PROJECTS THEORY APPLICATIONS CIRCUITS TECHNOLOGY

NUTS AND VOLTS

www.nutsvolts.com
November 2015

EVERYTHING FOR ELECTRONICS

Thinking of You

ToY with family and friends

Internet connected device says it all at the touch of a button

GPS COFFEE MAKER

TIMERS IN EMBEDDED SYSTEMS

OSCILLATOR TOPOLOGIES & APPS

Nano GPS click

November 2015

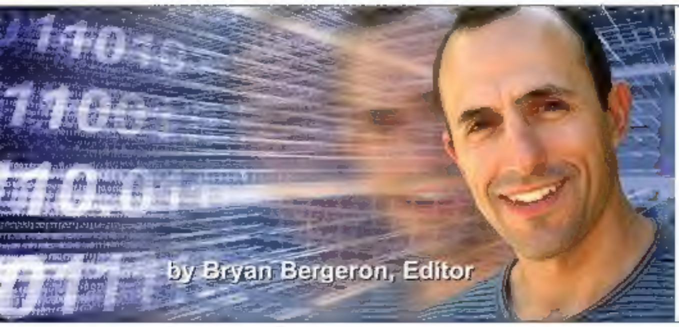
Page 44

Departments

Columns

Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

littleBits
GIZMOS & GADGETS



by Bryan Bergeron, Editor

DEVELOPING PERSPECTIVES

Wanted: Magnetic Free Zone

Magnetics, for the most part, make life easier. Consider what we'd do without the solenoids that actuate electric garage door motors, the rare-earth magnets embedded in iPad covers, magnetized tools, and the ubiquitous kitchen refrigerator magnets. However, the magnetic fields associated with magnets can be problematic.

For example, one of my interests is rebuilding vintage mechanical pocket watches. If you own a mechanical watch, you know that a magnetized watch will run abnormally fast. Well, I have a pocket watch on my desk that constantly gains time. I was at a loss to understand how the watch could become magnetized simply sitting on my desk. Well, using an inexpensive pocket compass, I was able to verify that the watch was being magnetized by a pair of scissors in a drawer directly under the watch. Opening and closing the drawer several times a day was enough to magnetize the watch – just as running a permanent magnet over a screwdriver can transform it into a magnet.

The discovery with my pocket watch led me to search for a magnetic free zone in my house. It was, in short, difficult. In my office, I have a dozen super magnets to hold papers on my white board. Then, there's the unshielded speakers on the wall. In my kitchen, I was surprised to learn that some of the flatware was magnetized. On my dresser, I found my steel collar stays and magnet sets. It seemed my compass never really settled on magnetic North, given the various motors and electronic gadgets around my place.

In retaliation, I purchased a few degaussing machines from eBay, where they can be had for about $10 and up. First up was the fixed magnet combined magnetizer/demagnetizer. These devices work great as magnetizers for long thin objects such as screwdrivers, but are useless in reversing the process.

Next, I tried the generic Chinese-built "blue box" demagnetizer – essentially an AC solenoid without the moving parts and a momentary on switch. You place the screwdriver or other object you want to demagnetize on top of the box and press the button, which energizes the core with 110 VAC. Then, you slowly move the object away from the unit as far as you can before releasing the switch. The iron molecules within the tool or other object should be randomly aligned, and therefore non-magnetic. This solution was affordable, reliable, and consistent.

Given that I was looking for a solution on eBay, I also had a serendipitous find – an old US made "instantaneous demagnetizer" tool by Magna Flux ($20). This tool uses a capacitor discharge to quickly ramp down the magnetic field after it's been built up. Like the blue boxes, it did the job. Moreover, there is no need to move the object to be demagnetized while the AC field is energized. Just press and release the button. The capacitor circuit takes care of decreasing the magnetic field.

Of course, if you decide to demagnetize your tools and mechanical watches, set up a safe area away from anything remotely resembling a magnetic data store. Don't think of using a demagnetizer around your credit cards or your DAT collection.

With the magnetics out of the way, I'm left to puzzle

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscription Orders
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ ADVERTISING SALES
Robin Lemieux
robin@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

VP OF OPERATIONS
Vern Graner
vern@nutsvolts.com

CONTRIBUTING EDITORS
Fred Eady
Tim Brown
Lou Frenzel
Andrew Retallack
Ron Hackett
Ward Silver

CONTRIBUTORS
Craig Lindley
Dave Prochnow
Ron Newton
Tom Bensky
Ron Hoffman

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

WEBSTORE MARKETING
Brian Kirkpatrick
sales@nutsvolts.com

WEBSTORE MANAGER
Sean Lemieux

ADMINISTRATIVE STAFF
Re Gandara

over why a mechanical watch would run faster when magnetized. Is it somehow more efficient because of decreased friction? Are Eddy currents somehow imparting energy to the mainspring? Could magnetized motors somehow run more efficiently? If you have the answer, please drop me a line. **NV**

READER FEEDBACK

Battery Blip

In the October 2015 Spin Zone column, it was incorrectly stated that the battery type for the badge was LiPo, when it's actually lithium-ion. Our apologies for any confusion.

(Not) All Wound Up

This missive is in reference to the article, *An Ultra Modern Shortwave Receiver* (July 2015) by George R. Steber.

The details of the reader-wound coils L5, L6, L7, L8, L9, and L10 do not specify wire size and type.

I rather imagine that enameled wire is used, but I do not know what gauge of wire.

G. Frank Humiston

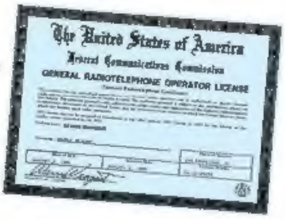
For the T50-6 cores, use #20 AWG wire. For the T68-6 cores, use #25 AWG wire. Use enameled type wire.

George Steber

Error Duly Noted

Regarding the Music Editing question/answer from the October 2015 Q&A:

Continued on page 78


The United States of America
Federal Communications Commission
GENERAL RADIOTELEPHONE OPERATOR LICENSE

Q & A

WITH TIM BROWN

In this column, Tim answers questions about all aspects of electronics, including computer hardware, software, circuits, electronic theory, troubleshooting, and anything else of interest to the hobbyist. Feel free to participate with your questions, comments, or suggestions. **Send all questions and comments to: Q&A@nutsvolts.com.**

- **Automotive Battery Amp-Hour Capacity at Slow Discharge**
- **Mailbag**

Post comments on this article at **www.nutsvolts.com/index.php?/magazine/article/november2015_QA.**

Automotive Battery Amp-Hour Capacity at Slow Discharge

Q What is the amp-hour capacity of a Size 24 automotive battery at a slow discharge rate of one amp or less? I know the cold cranking rating is 550 CCA.

— William Arford

A The amp-hour rating is — at least in theory — the amps that can be drawn from a battery for one hour, or the number of hours a battery can be used at a discharge rate of one amp. So, theoretically, a 200 amp-hour battery would last for 200 hours if discharged at a rate of one amp, or one hour if it were discharged at a rate of 200 amps — which would more than likely exceed the maximum discharge rate and destroy the battery in the process.

Amp-hour ratings are usually measured by discharging the battery over a 20 hour period. This would be great IF it were true for all discharge rates from maximum to minimum. However, in the "real" world, things are never so simple. As shown in **Figure 1**, as the rate of discharge decreases, the battery lasts longer and discharges to the endpoint voltage (point at which battery is unusable) over longer periods.

Cold Cranking Amps (CCA) is the number of amps a lead-acid battery can provide for 30 seconds at 0 degrees Celsius without the voltage between the positive and negative terminals dropping below around 15 percent (7.2 volts or 10.5 volts in battery specifications). The Size 24 automotive battery has a CCA of 600 to 800. Amp-hour ratings are for deep cycle batteries used in solar or wind power storage applications, for example, where the discharge rate is fairly low compared to cranking an engine; the battery may not be recharged immediately.

CCAs are used for automotive batteries that charge over a long period of time and discharge over a short period of time. Deep cycle batteries have thicker plates, denser active paste between plates, thicker separators, and may have different alloy materials than automotive batteries. So, the two are not compatible. There is no formula for converting CCAs to amp-hours since these are two very different types of batteries. It's like comparing apples and oranges.

To conclude, if you are looking at energy storage for a wind or solar project, use deep cycle rated batteries which will perform better in these applications than automotive batteries. **NV**

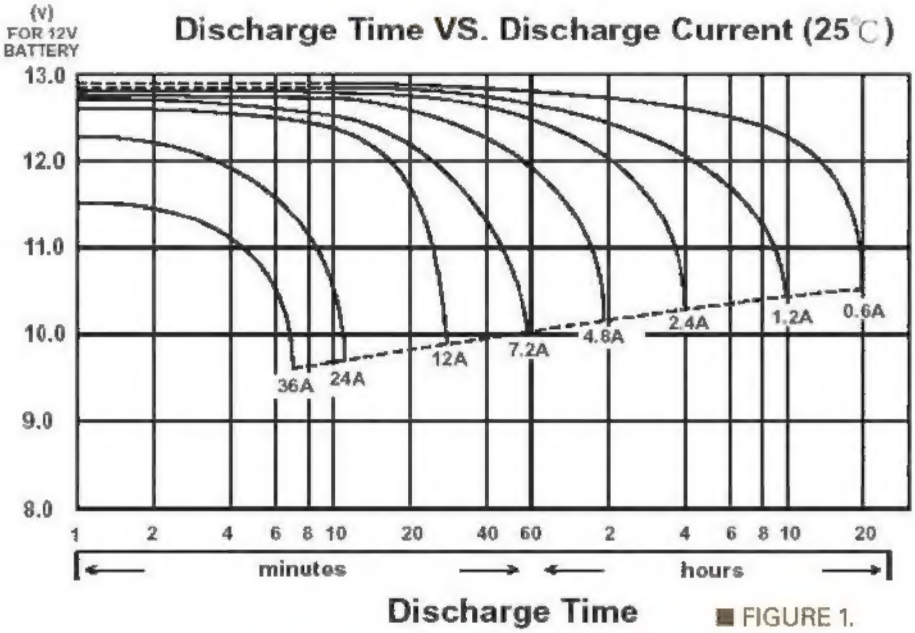


FIGURE 1.

MAILBAG

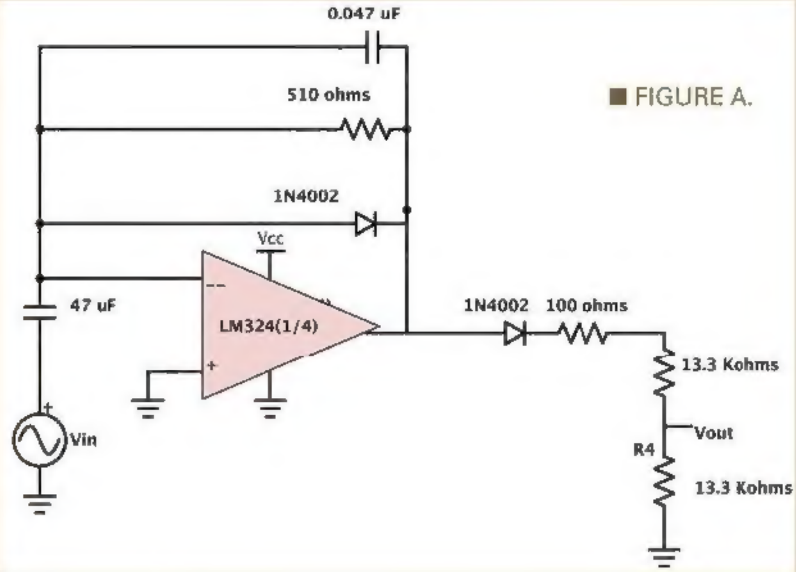
Re: May 2015 "Op-Amp Accuracy"

Would you please explain what Figure 1 in the May 2015 issue is supposed to do? I have never seen this circuit Also, Figure 2 will not work with the circuit in Figure 1. Figure 2 uses positive and negative power supplies, while Figure 1 uses only positive and ground.

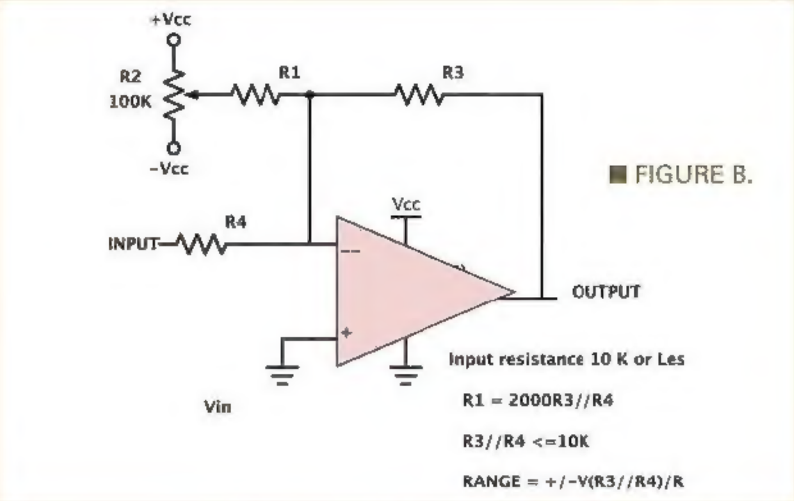
William Arford

*The May 2015 Figures 1 and 2 are shown here are **Figures A** and **B** for reference. The May Figure 1 (**A**) was an op-amp circuit provided by the reader who asked the question.*

*Figure 1 (**A**) is a precision rectifier circuit (a.k.a., super diode) with a 50 percent voltage reduction on the output with the feedback loop capacitor to remove output voltage spikes and the output resistance of the signal source as the op-amp's input resistance. When the signal source voltage is negative, the output is a replica of the input but amplified. When the signal source voltage is positive, the output is zero.*



■ FIGURE A.



■ FIGURE B.

Re: May 2015 "Q&A Tips Number 4"

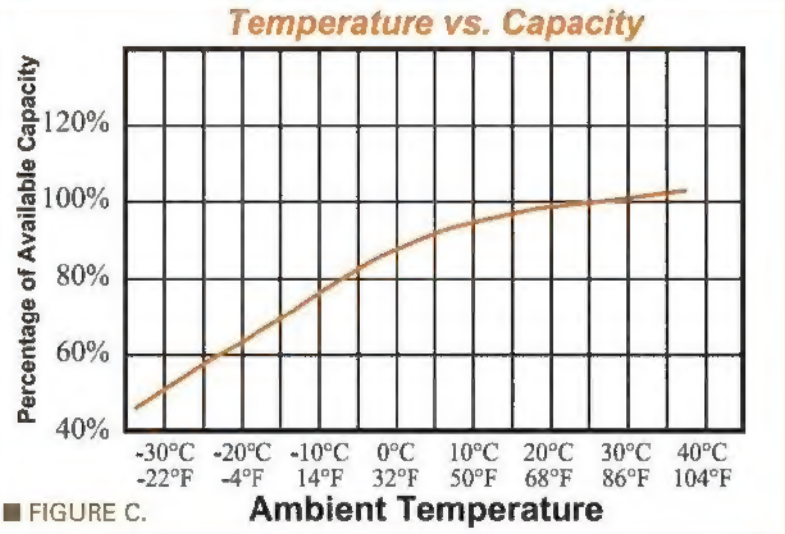
I know that you can keep batteries in the refrigerator, but I never heard of using the freezer. Is this okay? I purchased some Duracell Quantum AA cells marked for December 2023, so I question the need to freeze them. Fifty years ago, we did store batteries at 40 degrees Fahrenheit in the refrigerator to extend their life.

William Arford

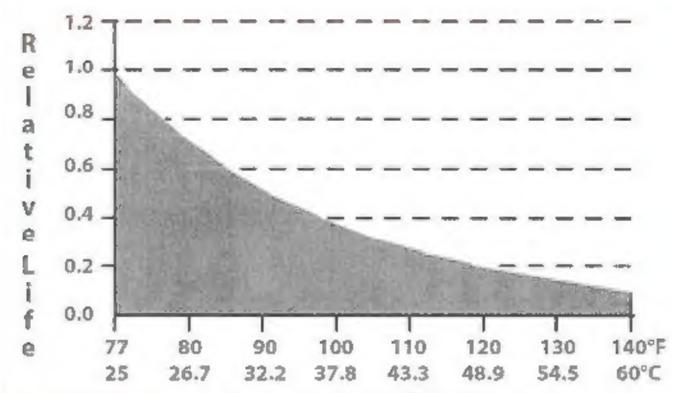
Storage of alkaline batteries in the refrigerator at 40 degrees will extend their shelf life. The reason for extended battery life at lower temperatures goes to the chemistry of the battery. Highly corrosive materials are used between electrodes made of two different materials (zinc anodes hooked to the negative terminal and magnesium oxide cathodes hooked to the positive terminal in alkalines) to generate roughly 1.5 volts.

*At room temperature, the chemicals are more active and thus produce the full battery capacity as shown in **Figure C**.*

However, as the chemical activity to generate voltage increases as temperature increases, so does the rate of corrosion of the battery's materials, so the battery's shelf life will decrease



■ FIGURE C.



■ FIGURE D.

*as shown in **Figure D**.*

The difference in your 40 degree storage temperature and my 0 degrees could mean you are getting around 1/3 of the shelf life I am getting (I am avoiding presenting the complicated math).

I have grandkids and we need batteries available constantly, so shelf life is a must for me. I can warm the batteries up very quickly by placing them close to my body if the TV remote or toy batteries go out.

As far as expiration dates printed on the batteries, I don't pay them too much attention. I have had batteries go "bad" that were still in the package, way before the expiration date. I have also had batteries still be good after the expiration date.

One special warning: Be sure to check batteries often when they are installed in your devices. I recently tried to use my programmable calculator (read VERY expensive) to sadly find the batteries were dead. The real problem was there was a clear skin-burning liquid inside the battery compartment which I removed post haste.

If you have ever tried to clean the battery corrosion products from a device, you know what I mean. It is a lot easier to check the battery condition periodically than to clean up the mess.

Re: July 2015 "GFI Breakers"

#1 I would suggest retesting the problem outlet except for the fact that the hair dryer worked, and at 1,200 watts it would draw 10 amps; meaning that the other wiring would get hot if it had, say, two ohms because it would use 20 watts, and thus get as hot as a small light bulb if left on for three to five minutes. Measuring the voltage at the remote outlet is a good idea, but my feeling is that this would be of no help because the two computer power supplies did not like a GFCI in the past.

What the computer power supplies may have had in common with the radio is a switching power supply. This seems like a bad idea at first, but could be a good thing. My eight year old CD player/radio picks up noise from my overhead LED light; this noise may be coming through an analog power supply. A properly designed switching supply would have very good filtering of 0 to 1 MHz so as not to pass anything to the audio section. Therefore, I would look for a difference in RF properties between outlets.

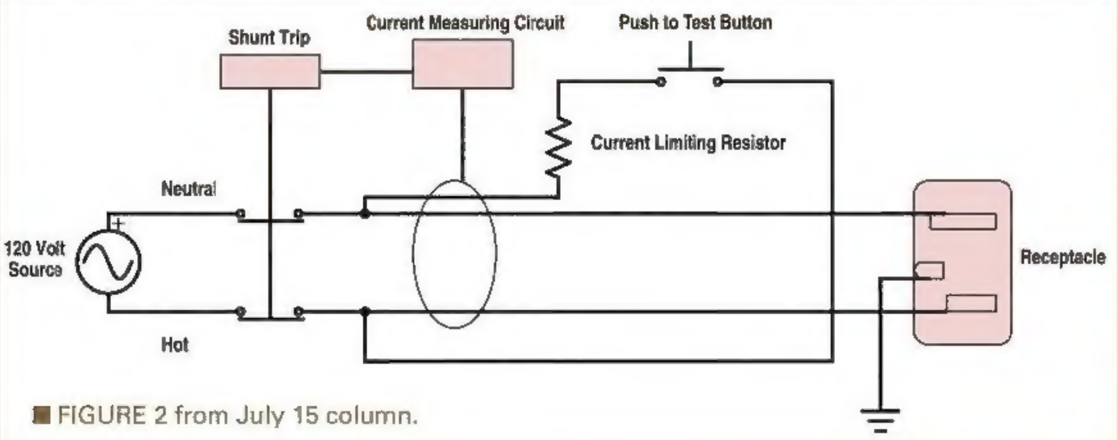
I would set up a scope to read the voltage at a normal outlet and plug the radio in. If the unit has a switching supply, some waveform may show up. Then, do the same on the protected outlet. You may not see the same wave form. If so, you have proven that the switching power supply does not like the GFCI due to some factor such as series inductance. The radio may also have some tricky defect in it that only shows up when using the GFCI, so as a last resort, buy another radio of the same kind and inspect its circuits to see what clues they offer.

William Arford

#2. A few points regarding the GFI Breakers question:

1) Mr. Edwards said that he was "changing plugs," but I suspect he meant that he was replacing the outlets.

2) Between the late '60s and late '70s, some homes were wired using aluminum wire. This might be contributing to the problem that Mr. Edwards described,



■ FIGURE 2 from July 15 column.

*A special note of appreciation ... I wrote the first part of this month's column six days after having a triple coronary artery bypass surgery. I would like to thank God, the medical professionals, family, friends, and the **Nuts & Volts** community for the support I have received during this time.* *Tim* since aluminum oxide can cause some very high resistance.

3) Mr. Edwards mentioned that this was an older home. My brother and I co-own a home that was built in the late '60s. While it had copper wire, several of the outlets either had evidence of arching (blackish residue on the faceplate) or had a significant voltage drop (about a 15V-20V drop under load). Upon closer investigation, I found that either (1) the screw terminals had loosened over time; or (2) that the copper wire had developed an oxide layer, causing additional resistance. I ended up replacing ALL of the outlets, and adding GFCI outlets in the kitchen and bathrooms. I also stripped back the insulation on all of the wires to get clean copper. Problems solved!

4) Without doing ANY investigation (aside from measuring the voltage at the outlets), an electrician wanted to charge me $4K just to pull new copper wire and replace the breaker box. While I haven't (yet) replaced the breakers, I saved a bunch by replacing the outlets myself. I think the existing copper wire is fine, and I'll replace the breakers as time permits.

Jeff Kerner

#3 I'm not sure where the diagram (**Figure 2**) shown in the July magazine of the GFI originates, but the test button will not work. To test the GFI, it must simulate an earth leak, and the resistor and test button should therefore be between the "hot" and "ground." As drawn, it can pull the full rated current of the GFI.

Bill van Dijk

#4 I bet Mr. Edwards GFI problem was caused by RFI. Mr. Edwards did not indicate if the radio had a wall wart power supply, but if it did, the RFI from the wall wart can interfere with some GFI designs. I have a wall wart on my security camera that generates RFI. My scanner picks up the RFI at about 120 MHz in the aircraft band, with more spurs extending into the two-meter ham band. Clip-on ferrites minimized the detected interference to a few feet from the wall wart. My wall wart is not on a GFI outlet. I have read reports from hams about GFI breakers tripping when a handheld transmitter is used nearby. GFI breaker designs are changing every year to cut costs and improve reliability, and all the possible application issues may not be well understood by manufacturers.

Leroy Higgins KI6MES

#5 I would like to offer a suggestion on the GFI issue. The writer doesn't state whether the radio doesn't power up at all or that it just doesn't play. I could see one issue making it not play while plugged into a GFI. The GFIs use a comparator since just about all inexpensive table radios use the AC cord as an antenna. Any noise generated by the comparator circuit might be obliterating stations on the FM band and bringing noise into the radio to do the same on the AM band. If it doesn't power up, well, this theory is out the window.

Ron Schacht

*Good advice, Dale, Jeff, Bill, Leroy, and Ron! **Figure 2** in the July issue is a standard GFI schematic. In normal operation, the same current flows through the "hot" and "neutral" wires, thus the magnetic fields cancel out in the measuring loop (or coil) and the GFI breaker remains closed unless a ground fault current is diverted through, say, a human being and the currents are not balanced, which trips the GFI breaker portion of the circuit.*

When the test button is pushed (in this schematic), all of the current flows through the sensing loop test branch (simulating flowing through the "neutral" branch). However, no current flows through the "hot" branch (simulating a ground fault and tripping the GFI breaker). I should have done a better job of explaining the test portion of the GFI schematic.

BY RON HACKETT
www.jrhackett.net

PICAXE-PC Serial Communication — Part 4

If you've been following our PICAXE-PC Communications articles in the Primer, you know that we're now at the point where we're ready to move on to using a "master program" on the PC to communicate with a PICAXE project, and that we're going to use Python as the programming language to do so. However, I do realize that many Primer readers would prefer to not get involved in further Python-related projects. So, this month, we're going to begin by using three different approaches to transferring data from a PICAXE project to a PC, so that it can be stored on a PC and then displayed in a Microsoft Excel chart. The first two will not require Python at all, but the third approach will involve a simple Python program, so you can choose whichever approach you prefer.

All three experiments will use the same small set of sample data that I chose to illustrate the techniques involved in the processes of transferring the data to a PC and displaying it in an Excel chart. The sample data is small enough to easily be included in the PICAXE data transfer programs that we'll be using here. However, many "real" data-logging projects need to be able to collect and store several kilobytes of data internally, before that data is transferred to a PC. Since the storage capability of the PICAXE M2-class processors is somewhat limited, the usual solution to this problem is to include an external EEPROM to increase the storage capability of the project.

Back in the August 2011 column, we experimented with using an EEPROM as a non-volatile storage device. However, technology has advanced considerably since then, and now there are reasonably priced Ferroelectric-RAM (FRAM) chips that can accomplish the same purpose. Like EEPROMs, they provide non-volatile storage but they are much faster than an EEPROM. Plus, they are much easier to work with because they don't have any of the page-boundary complications associated with page-structured EEPROMs. In other words, we can sequentially store any number of data bytes in an FRAM, as long as we don't exceed the total capacity of the chip.

In our fourth and final experiment this month, we'll see how easy it is to interface a 32 kByte FRAM with an 08M2 processor, and explore the process of FRAM data storage and retrieval. By the time we're finished, we will have covered everything we

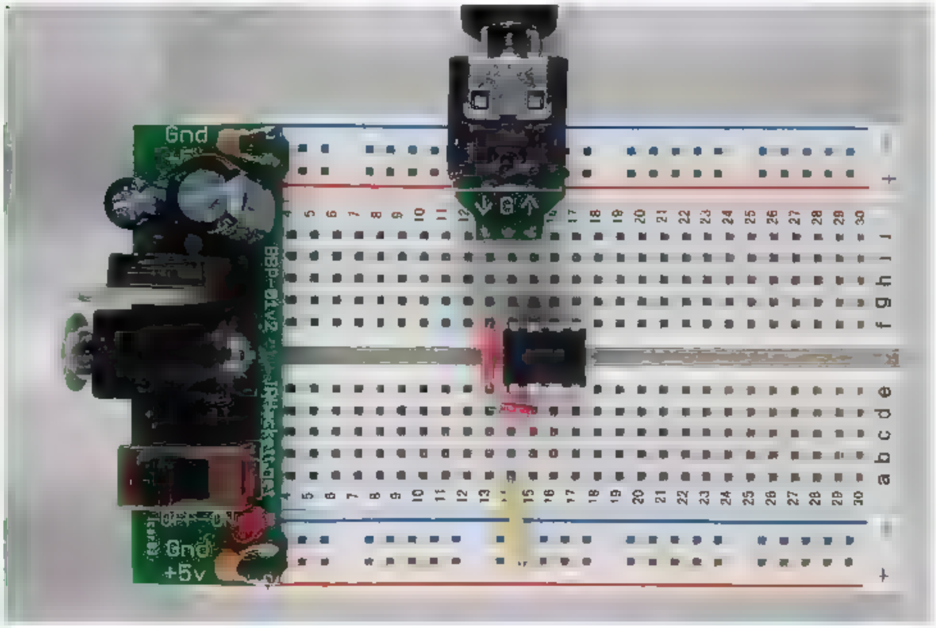


FIGURE 1. Breadboard setup for Experiment 1.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/November2015_PICAXEPrimer.**

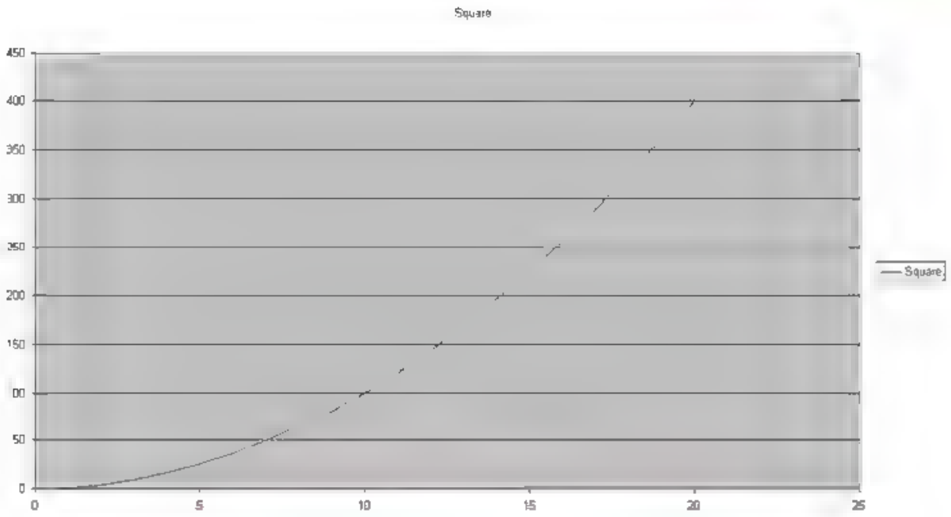
need to implement a simple 08M2-based data-logging project, which will be the topic of the next installment of the Primer.

Experiment 1: PICAXE to PC Data Transfer (Version 1)

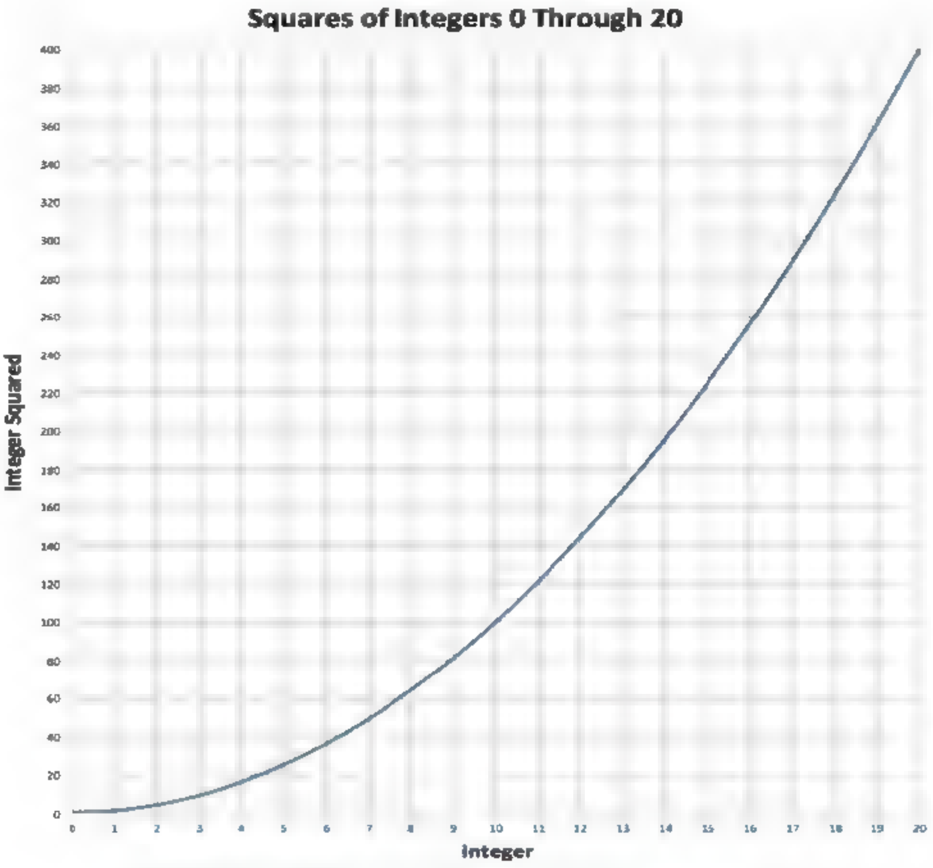
The hardware setup for Experiment 1 is probably the simplest we've ever used: an 08M2 processor, a 5V power supply, and a programming adapter. (My breadboard setup is shown in **Figure 1**.) The software for Experiment 1 (`Sertxd2Excel.bas`) can be downloaded from the article link, along with the other programs that we'll be using this month. When you've downloaded the programs, open `Sertxd2Excel.bas` in the PICAXE Programming Editor and read through the program listing. Essentially, it's just a series of 22 `sertxd` statements that transmit data to the terminal window. As we'll soon see, the first serial transmission (`"Integer,Square",cr,lf`) will be used to specify the column headings in Excel. Each of the remaining transmissions sends the following data: an integer, the "comma" character, the square of the initial integer, and the ASCII `cr` and `lf` characters. The important thing to note is that all the data is transmitted in ASCII format.

Don't forget, whenever a constant or variable is preceded by the "#" symbol in a serial output statement (`serout`, `hserout`, or `sertxd`), the value is transmitted as a string of ASCII characters. For example, a value of 196 would be transmitted as the following three ASCII values: 49, 57, and 54, which represent 1, 9, and 6, respectively.

When you're ready, download the `Sertxd2Excel.bas` program to your breadboard setup. All the transmitted data should appear in the terminal window almost immediately. The next step is to save the transmitted data in a file. Exactly how you do that depends on whether you've upgraded to version 6 of the PICAXE Editor. If you are using version 6, simply click the "Save" button near the bottom right corner of the terminal window, name the file "Data.csv," and save it as a text file on your desktop. The "csv" file extension stands for "comma separated values."



■ **FIGURE 2.** Initial Excel chart for Experiment 1.



■ **FIGURE 3.** "Improved" Excel chart for Experiment 1.

This type of file can be opened by Microsoft Excel. As you're saving the file, you might get a warning that says your file may become unusable — just ignore it. If you get a second warning that another program is using the file, you will need to close the terminal window to proceed.

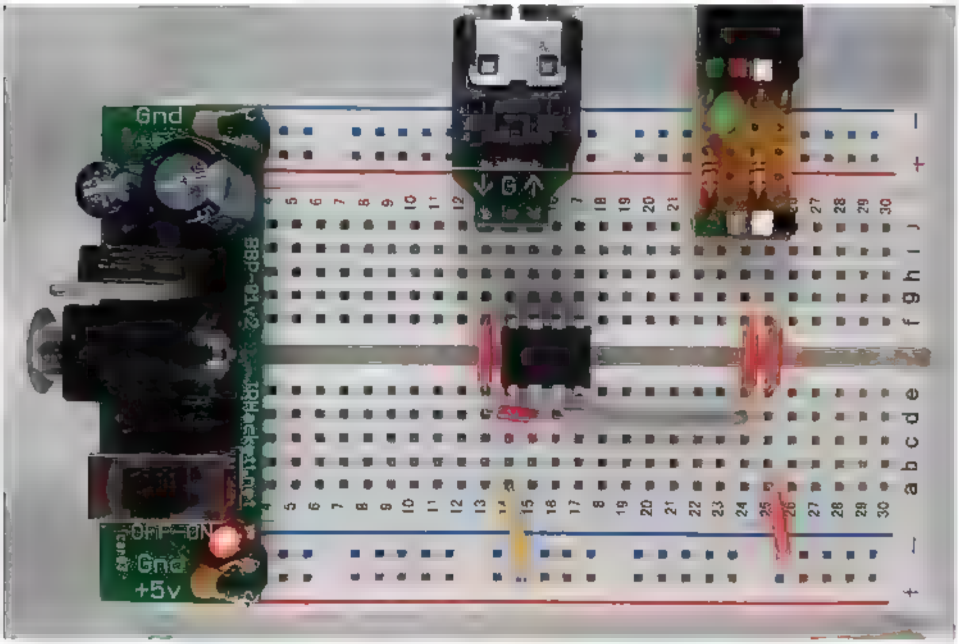
On the other hand, if you're using version 5 of the PICAXE Editor, select the "Edit" menu, and choose "Copy Input Buffer." Now, open the WordPad application and paste the data into WordPad. Finally, select "File" → "Save As..." from the menus, name the file "Data.csv," and save it as a "text document" (**not** the "MS-DOS Format" option) on your desktop. Again, just ignore any warnings you receive.

When the file has been saved on the desktop, its icon should include a small Excel icon; double-clicking the file icon will open the file in Excel, with the comma-separated data appearing in two contiguous columns (without the "comma" in between).

The next step is to create an Excel chart from the data. To do that, simply select all the data (columns A and B, rows 1 through 22), and click the "Insert" → "Chart" menu item. I chose the "XY (Scatter)" option with smooth curves. (You may want to experiment with other chart variations.) Your chart should look something like what is shown in **Figure 2**. Actually, it probably will look a lot better, because I'm still running Excel 2002 under Windows XP as a virtual machine on my Mac.

My OS X version of Excel is much newer (2011), so I followed a similar procedure to produce the chart again. This time, I took advantage of Excel's chart editing capabilities, and produced the "improved" chart that's presented in **Figure 3**. We won't discuss the details of editing Excel charts because I'm sure they differ somewhat between OS X and Windows. Besides, you can always ask Coogle for help – I certainly did!

;Experiment 1 is a simple demonstration of one way that a PICAXE program can send data to a PC for storage and analysis. In Experiment 2, we'll investigate a second option for accomplishing the same tasks.



■ **FIGURE 4.** Breadboard setup for Experiments 2 and 3.

Experiment 2: PICAXE to PC Data Transfer (Version 2)

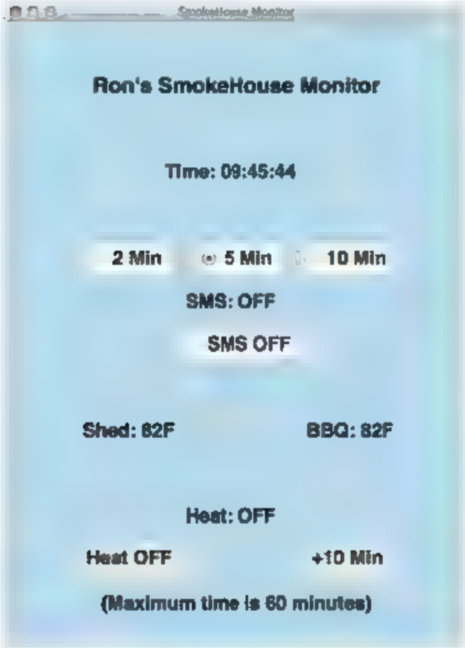
One disadvantage of the approach we just used in Experiment 1 is that it's necessary to have the Programming Editor running whenever you want to capture data for an ongoing PICAXE project. In itself, that's not much of a problem. However, it also means that you can't work on any other PICAXE project while the Terminal Window is busy monitoring incoming data. The solution to this problem is to use the same approach we experimented with in Part 1 of our PICAXE-PC Communication series. We can connect one end of a Prolific PL2303HX USB-to-serial cable to any output pin on a PICAXE processor, insert the other end of the cable into an available USB port on the PC, and use a separate terminal program (e.g., CoolTerm) to capture and store the data in a text file.

My breadboard setup for Experiment 2 is shown in **Figure 4**. As you can see, I'm using the stripboard interface we constructed for the PL2303HX cable in Part 1 of this series, but you can directly connect the cable to your breadboard circuit if you prefer. Just make sure you connect pin C.4 on the 08M2 to the cable's `RxD` pin, which is the one that's painted white on my interface board.

The software for the experiment (`Serout2Excel.bas`) is functionally the same as the program we just used, except that all the `sertxd` statements have been changed to `serout` statements because we're no longer using the `Serial Out` pin (C.0) on the 08M2. Also, note that I added a high `TxPin` statement and a `pause :100` statement. These two statements configure the `TxPin` in a high state, which is what is required for "true" (non-inverted) serial communications.

When you've completed your breadboard setup for Experiment 2, the next step is to determine which serial port the Prolific cable is using. As we did in Part 1, just run your terminal program and look at the list of ports the program sees. (In CoolTerm, that list is available under the "Options" → "Serial Port" menu.) Next, plug in the Prolific cable, re-scan the serial ports, and look at the list again; the newly added port is the one that's connected to the Prolific cable. After you've selected the correct serial port, configure your terminal program for 9600 baud (8-N-1) with no flow control.

At this point, the `Serout2Excel.bas` program is not yet running on your breadboard circuit. As a result, if you connect CoolTerm to the PL2303HX port, you will probably get a "Break Condition" error. (The same may be true for other terminal programs, as well.) A "break condition" occurs when the serial line is not idling in a "high" state (for "true" or non-inverted serial communication). We could avoid this problem by downloading the `Serout2Excel.bas` program to the breadboard setup before running our terminal program, but (of course) the terminal program would fail to receive any of the data because it would have been sent before we activated the terminal program.



Ron's SmokeHouse Monitor

Time: 09:45:44

SMS: OFF

Shed: 82F
BBQ: 82F

Heat: OFF

(Maximum time is 60 minutes)

■ **FIGURE 5.** Python GUI for PICAXE program control.

The way to avoid this catch-22 situation in CoolTerm is to select the "Options" → "Receive" menu, and click in the check box for "Ignore receive signal errors." (If you're using a different terminal program, hopefully it includes a similar option.)

When you've correctly configured your terminal program, connect it to the PL2303HX port, and then download the `Serout2Excel.bas` program to your breadboard setup. The terminal program should display the same text we saw in Experiment 1. To save the data to a file, choose "Select All" from the Edit menu, and then choose "Copy" from the Edit menu. Next, open WordPad, and paste the data into WordPad. As we did in Experiment 1, select "Save As..." from the File menu, name the file "Data.csv," and save it as a "text document" (**not** the "MS-DOS Format" option) on your desktop. Again, just ignore any warnings you receive.

As before, when the file has been saved on the desktop, its icon should include a small Excel icon, and double-clicking the file icon will open the file in Excel with the comma-separated data appearing in two contiguous columns (without the "comma" in between). Naturally, you can again display the data in an Excel chart if you want.

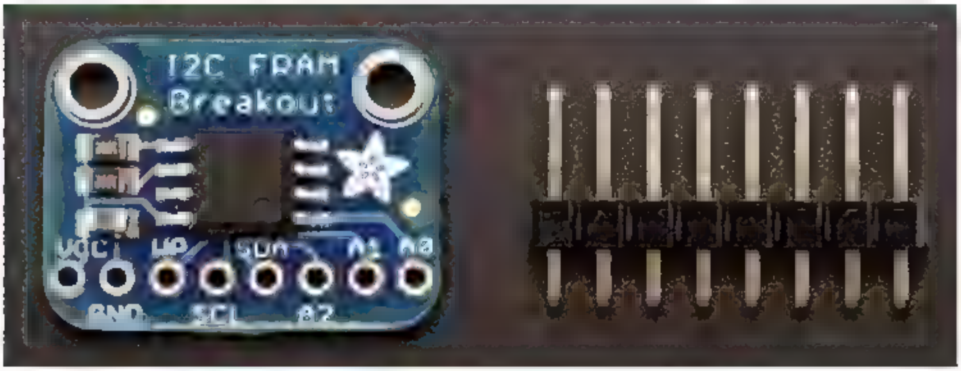
Experiment 3: PICAXE to PC Data Transfer (Version 3)

In Experiment 3, we're going to replicate the goals of our first two experiments, but this time we're going to use Python to receive and store the incoming data. Before we get started, I want to clarify an important point. If data collection and storage are your only goals, you don't need Python at all. The approach we just used in Experiment 2 is more than adequate for those tasks. However, the real power of Python is that it's capable of much more.

To begin with, Python is cross-platform, so a program developed for one platform can easily be ported to another. In addition, one of Python's most significant features is its ability to provide a GUI (graphical user interface) for a running PICAXE project. For example, when I was developing the PICAXE-Pi series for the Primer, I wrote a Python program to enable a Raspberry Pi to monitor an electric smoker in my outdoor shed, and to control a small electric heater for my own comfort in the winter. (I got tired of going out to a freezing shed to turn on the heater, so now I can do that from my Mac.)

As I was writing this article, I became curious to see whether the GUI program I wrote for a Pi would run on my Mac. At first, it didn't, but all I had to do was comment out any reference to the Pi's GPIO pins, and then the program ran fine (see **Figure 5**). Now, I just need to find the time to develop a PICAXE project that communicates with the master Python program on my Mac!

The breadboard setup for Experiment 3 is identical to what we just used in Experiment 2. The PICAXE program (`Serout2Python.bas`) is very similar to our previous programs, but it does contain two important differences. Open `Serout2Python.bas` in the Programming Editor, and let's start with the simpler of the two. At the bottom of the program, you can see that I've added an additional statement: `serout TxPin,T9600_8,("end of data",lf)`. This statement tells the



■ **FIGURE 6.** Adafruit FRAM breakout board.

companion Python program when to stop inputting data, close the data file, and exit the program. The second difference is that all the `serout` statements no longer include the "`cr`" character in the output string.

In my initial testing of the program, I did include the "cr" character, but I was never able to get Excel to correctly format the data. This change is necessary because Python expects only one character (the ASCII `lf` character) to mark the end of the line of data.

Now, let's turn our attention to the `Serin2File.py` program that we'll be using to receive and store the data sent from our breadboard circuit. If you have Python 2 installed on your PC, you can open the program file by first running the `idle` program for Python 2. If not – and you just want to view the program listing – you can open it in WordPad. From the "File" menu, choose "Open," and then specify all files; click on the `Serin2File.py` file. When you read the program listing, you'll see several Python statements that we've already used in our PICAXE-Pi articles, so I won't discuss them further. However, the `with` statement is new to us, so a brief explanation is in order.

The Python `with` statement simplifies exception handling by automatically handling common programming errors. In other words, it eliminates the necessity of writing code that includes the `try…except` blocks that are typically used to handle common errors, such as a serial port not being available. The `with` statement also automatically takes care of common cleanup tasks, such as closing a port when we're finished using it. If you want more details, just search online for "Python `with` statement."

The comment following each line in the main `while loop` of the Python program should be enough to explain what each line is doing, but a couple of points are worth mentioning. First, the `while loop` is an infinite loop, but the `if` statement checks each line of serial input before the line is printed or saved to the text file. If the line contains the string "end of data," the program breaks out of the loop, prints the final statement, and terminates.

Also, it's worth noting that we can't write "`if newdata == 'end of data':`" because the line includes an invisible `lf` character. Finally, the trailing comma in the "`print (newdata),`" statement stops Python from adding a new line character to the print statement because there is already one in the line of data. If Python added a second new line character, the printout would be double-spaced.

If you have Python 2 installed on your PC and you want to run Experiment 3, just carry out the following steps:

1. Open the `Serin2File.py` program in the Python 2 `idle` Shell.
2. Open the `Serout2Python.bas` program in the PICAXE Editor.
3. Arrange the open windows so that you can see everything simultaneously.
4. Run the `Serin2File.py` program; it will wait for the first line of serial input.
5. Download the `Serout2Python.bas` program to your breadboard setup.

As soon as the PICAXE program has finished downloading to your breadboard setup, you should see the data appear in the `idle` window. If not, you will need to troubleshoot your setup!

FRAM Data Storage and Retrieval

Most FRAM ICs are available only in SMD packages. However, I realize that many readers aren't comfortable soldering surface-mount components, so we'll use a FRAM breakout board (product #1895) from **Adafruit.com** for our next experiment. The Adafruit board is based on the Fujitsu MB85RC256V 32 kByte FRAM which is an I²C device, so it's easily interfaced with any PICAXE processor. As you can see in **Figure 6**, the breakout board is fully assembled, except for an (included) eight-pin straight male header which can be used to insert the breakout board into a breadboard. The relevant details of the eight pins (from left to right) are as follows:

- **Power Pins (Vcc and GND):** The operating voltage can range between 2.7V and 5.5V, so the FRAM board can be used in either 3.3V or 5.0V projects.
- **Write Protect Pin (WP):** This pin is internally pulled low (Write

Enabled) in the FRAM chip. To Write-Protect the FRAM, you can just connect Vcc to the `WP` pin. I have no need to write-protect the FRAM, so I didn't connect the `WP` pin at all. (More about that shortly.)

- **I^{2}C Pins (SCL and SCA):** These are the standard I^{2}C pins (Serial Clock and Serial Data). They are each held high by a 10K SMD resistor on the breakout board. I was initially concerned about this because the PICAXE I^{2}C documentation specifies 4.7K pull-up resistors. However, the board seems to function perfectly with an 08M2 processor.
- **Address Pins (A2-A0):** These three pins are internally pulled low in the FRAM chip, so the default seven-bit I^{2}C address is 0x50 (%1010000). By applying `Vcc` to any combination of these pins, the I^{2}C address can be configured anywhere between 0x50 and 0x57. Again, I have no need to change the I^{2}C address of the FRAM, so I also didn't connect the three address pins at all.

At this point, I should explain why I chose not to connect four of the eight pins on the Adafruit FRAM breakout board. In the next installment of the Primer, we'll get into the details of the specific data-logging project I have in mind. For now, I'll just say that my plan is to implement what used to be called a "sneaker-net" approach to data collection. I want to configure a data-logging circuit so that it can store a large amount of data in the FRAM board over an extended period of time, at a location that is "remote" (i.e., not near my Mac). When the data collection is complete, I want to be able to remove the FRAM from the data-logging circuit, physically carry it to my Mac, and then insert it into a second breadboard circuit that will serially send the data to the Mac so I can view it in an Excel chart.

To simplify that process, I want it to be as easy as possible to remove the FRAM board from one breadboard circuit and insert it into another breadboard circuit. Fortunately, for my purpose, the pin-out of the FRAM board makes it an ideal component in an 08M2 circuit. **Figure 7** presents the necessary interface connections.

As you can see, the removal of the four pins (`WP, A2, A1,` and `A0`) makes it possible to insert the FRAM board directly in line with the pins on one side of the 08M2 because the 08M2 pin C.1 is its SCL connection, and the 08M2 pin C.2 is its SDA connection. Finally, only one jumper wire is required to connect `Vcc` to the FRAM breakout board which further simplifies the interface.

In addition to removing the four pins from the board, I also used a right-angle male connector, rather than the straight connector provided with the FRAM board (see **Figure 8**). This way, the board is inserted and removed vertically, which makes it much easier to move between two breadboards.

Experiment 4: Testing the FRAM Breakout Board

In preparation for the next column, let's test the FRAM board to make sure it functions correctly. The breadboard circuit we'll use is presented in **Figure 9**. As you can see, it's necessary to add a jumper to the +5V rail for the `Vcc` connection on pin 1 of the FRAM board. Also, note that it's necessary to offset the PICAXE programming adapter from its normal position to avoid a conflict between the +5V connection to pin 1 on the FRAM board and the `Serin` connection on the program adapter.

The program we're going to use (`FRAMtest.bas`) simply demonstrates several different ways of writing data to and reading data from the FRAM. Before we actually run the program, there are a couple points that I want to clarify, so this would be a good time to open `FRAMtest.bas` in the PICAXE Editor and use it for reference during the following discussion (either on your PC's screen or on a print-out of the program listing).

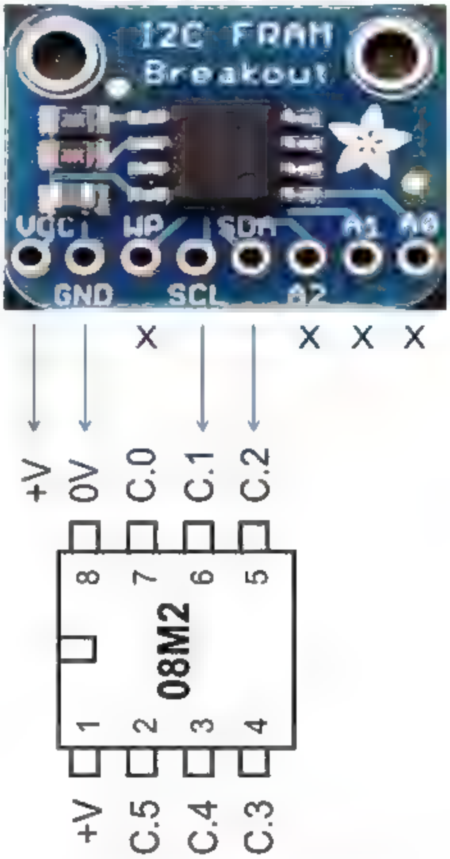
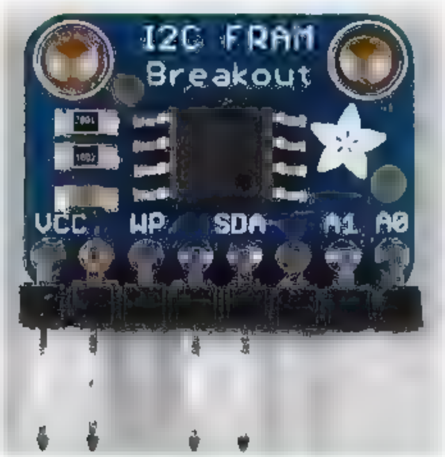


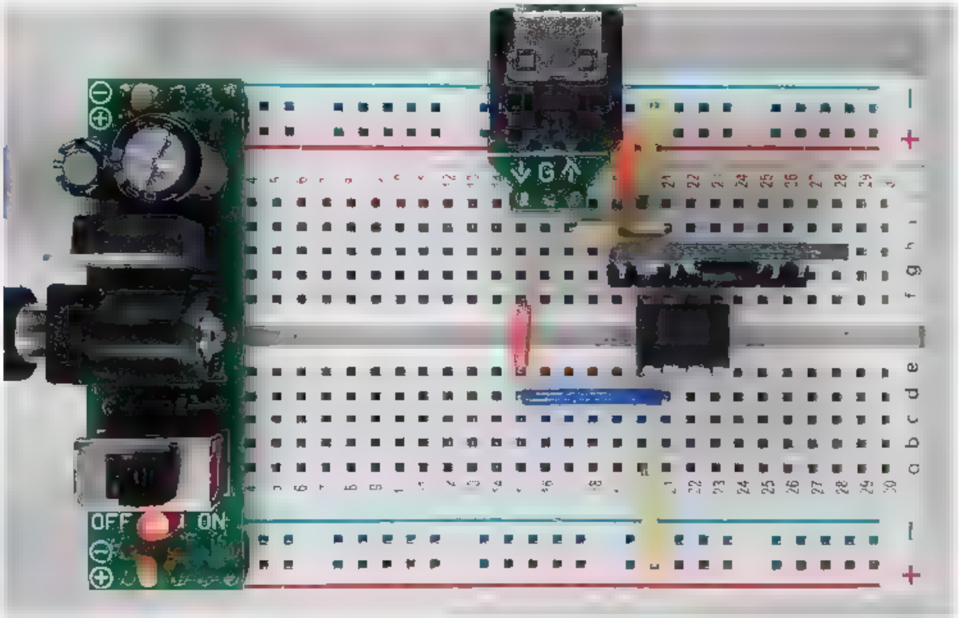
FIGURE 7. FRAM interface connections for PICAXE 08M2.

The most important point to make is that we've already discussed the details of PICAXE I^{2}C communication in the August 2011 installment of the Primer ("Using EEPROMs to Store and Retrieve Data"), so this month, we're only focusing on the differences between EEPROM and FRAM storage. (Before reading further, it might be a good idea to review the I^{2}C basics we covered in the EEPROM article.)

I also want to clarify a possible



■ **FIGURE 8.** Completed FRAM breakout board.



■ **FIGURE 9.** Breadboard setup for Experiment 4.

source of confusion. If you read the documentation that's available at **Adafruit.com**, you'll see that the I²C address of the FRAM breakout board is listed as 0x50 (50 hex, or 1010000 binary), but in the `FRAMtest.bas` program listing, we're using 0xA0 (A0 hex, or 10100000 binary). So, an explanation is definitely in order!

To clarify this apparent inconsistency, we need to examine the FRAM's complete eight-bit address which is required in the PICAXE `hi2csetup` command (see **Figure 10**). As you can see, the high nibble of the complete address is %1010. This value is assigned to all FRAM (and EEPROM) devices by the I²C protocol, so every FRAM has this portion of its address already stored internally. The voltage levels we set for the FRAM's A2, A1, and A0 pins determine bits 3, 2, and 1 of the FRAM's `slaveaddress`. Since we aren't applying `Vcc` to any of these three pins, those bits are all equal to 0.

Bit 0 is where the confusion begins. It's the `read-write` bit, and it's automatically set high for a read operation and low for a write operation. Both the PICAXE and the FRAM ignore the value we place in bit 0 of the eight-bit `slaveaddress`. They automatically manipulate it as necessary during read and write operations. Since it doesn't matter what value we place in bit 0, I have set it to 0 which yields a full eight-bit `slaveaddress` of `%10100000` (A0 hex). The problem is that the higher seven bits are frequently referred to as the I²C "address" of the device which is technically correct, but we need to include the `read-write` bit in a full eight-bit address (even though it's ignored anyway). So, we need to use %10100000 (A0 hex) and not just the highest seven bits (%1010000, or 50 hex) in our programs.

With that confusion hopefully out of the way, let's focus on the four different variations of I²C writing and reading that are illustrated in the `FRAMtest.bas` program. The number of each of the following comments refers to the corresponding number in brackets preceding each comment at the left edge of the program listing:

[1]This is — by far — the simplest example; a single value is stored at location 100 in the FRAM, then the value is retrieved and displayed in the terminal window.

[2]When a series of values are stored in a single `hi2cout` command, storage begins at the specified FRAM location, and then the location is automatically incremented for the storage of each subsequent character. In the first `for/next` loop, prefixing the "#" symbol to the `var` variable results in the `sertxd` statement sending each individual digit to the terminal, so the actual values are displayed. The second `for/next` loop — which does not include the "#" prefix — sends each value as a single byte, which the terminal displays as the appropriate ASCII character.

[3]This example demonstrates how to send a string of ASCII text, and then retrieve and display it as the original ASCII text (because the "#" prefix is not included in the `sertxd` command). The `for/next` loop "counts" the number of characters in the text string, so that the original string is accurately retrieved and displayed in the terminal.

[4]The final example demonstrates how a text string can be retrieved and displayed without needing to know the number of characters in the string. Here, we're using

1	0	1	0	A2	A1	A0	R/$\overline{W}$

■ **FIGURE 10.** Complete eight-bit FRAM address.

the ASCII "end of text" character (`etx`), which has an ASCII value of 3. The `if/then` statement tests each retrieved character. If it's the `etx` character, then the `exit` command immediately terminates the `do/loop`. We can actually use any value we want for this purpose, as long as it's between 0 and 31. The reason is that ASCII values 32 and above represent printable characters. If we were to use a value greater than 31, a character in our text string could prematurely terminate the retrieval process.

There's one final point (unrelated to our upcoming data-logging project) that I want to mention. We're currently focusing on storing and retrieving real time data so that it can be charted in Excel on a PC. However, the technique we just discussed in the final example of the `FRAMtest.bas` program can also be used to transmit stored data to an LCD. As a result, FRAM storage can be a powerful technique for any PICAXE project that relies heavily on displaying data on an LCD.

For example, we could write a PICAXE program that stores every phrase we could conceivably want to display on the LCD (terminated with an "end of text" marker) in the FRAM, and note the starting location of each phrase. A fairly complex program could require a significant amount of stored data, so including a FRAM in such a project would free up a considerable amount of memory in the PICAXE processor.

We're once again out of space, so that's it for now. Next time, we'll use what we've learned about PICAXE-PC communication and FRAM data storage to develop an 08M2-based data-logging project.

See you then ... **NV**

EMI/RFI: The Bane of Wireless

Practically every electronic device is an interference generator and susceptible to interference. Think about this: We are all surrounded by a massive invisible cloud of wireless signals and electromagnetic fields. Wireless or radio, of course, is electromagnetic waves for communications applications. All of these radio signals in the universe are in the mix, as well as electromagnetic fields generated by other devices for non-communications purposes. This huge nasty brew can be benign or cause errors or disruption of some kind. We are all contributors to and targets of this mess of signals. Here is a brief introduction to this natural phenomenon.

First Some Definitions

Just to be clear on what we are talking about here, let's define a few often-used terms. The main term in use is electromagnetic interference (EMI). EMI is an electromagnetic signal from any source that can disrupt the operation of some electrical or electronic device. EMI may introduce data errors, garble voice or video signals, or totally prevent the normal operation of a circuit or device. EMI is also called radio frequency interference (RFI). RFI is the disruption to any wireless or radio communications device such as two-way radios, TV, satellite, or radar.

Another term you will hear is electromagnetic compatibility (EMC). EMC is the state achieved where electrical or electronic devices do not interfere with one another or themselves. It is a condition where EMI emissions are small and controlled, and products are made to be immune to the interference. It is an ideal state that is the goal of all electronic manufacturers. Nothing is perfect, but government regulations are in place to help everyone achieve this state.

EMI Sources

There are many different types of EMI and ways to categorize it. For example, there is intended and unintentional interference. Intended transmission represents signals generated for communications purposes by radio and TV broadcast stations, two-way radios, cell phones, wireless LANs, satellites, and radar. Some of these signals could interfere with other devices.

Unintentional EMI may derive from harmonics or intermodulation products of intended signals. Or, they could come from nearby computers or other digital equipment. Other sources are the AC power line or the atmosphere.

EMI is either man-made or natural. Man-made EMI is generated by electronic circuits. Examples are local oscillators or VCOs in receivers, oscillators in transmitters, clocks in computers, or other digital equipment. Digital circuits are major generators of EMI since their fast rise time signals are generators of harmonic energy. One massive EMI source is switching power supplies like switching regulators, DC-to-DC converters, and DC-to-AC inverters. Other man-made noise comes from things like auto ignitions, arc welders, switching AC power line loads off and on, and electrostatic discharge (ESD).

As for natural EMI, a great example is lightning and other atmospheric sources. Cosmic noise from the stars is EMI. The sun is a noise or EMI source. So are magnetic storms like the aurora borealis

These types of EMI may be continuous or intermittent or pulsed. EMI from a digital clock is continuous, as is the RF from a microwave oven. An example of a pulsed source is the transient voltage and current that occurs when a load like a motor is switched off or on to the AC power line. A massive EMI generator is a compact fluorescent (CFL) light bulb that uses a switching power supply operating in the RF range.

Of course, you may have a mix of these conditions. Think of the EMI in the well used license-free industrial-scientific-medical (ISM) band from 2.4 GHz to 2.5 GHz. It is used by a wide range of devices such as any Wi-Fi router or connected device, Bluetooth devices, cordless phones, ZigBee wireless sensors, printers, video drones, and microwave ovens. There are millions of such devices sharing that spectrum. It's a miracle that they work at all.

Post comments on this article and find any associated files and/or downloads at www.nutsvolts.com/index.php?/magazine/article/November2015_OpenCommunication.

Propagation Methods

Just how does EMI get into the "victim" device anyway? Actually, there are two ways: conducted and radiated. Conducted EMI is sent to a device via some wire or cable or other direct connection. One common way is via the AC power line. Switching spikes from loads are a nuisance and other noise on the line. Even power line communications (PLC) systems that use frequencies up to 30 MHz to transmit data on the AC mains can become EMI to someone. EMI in the form of power supply ripple and noise — especially from switch-mode power supplies (SMPS) — can be passed to another circuit or unit by way of the DC power bus.

Radiated EMI gets into a device by way of coupling via mutual capacitance, induction, or radio waves. Remember that any current-carrying conductor creates a magnetic field that can induce a voltage into any nearby conductor. In addition, signals in one conductor can be coupled to another conductor by mutual capacitance.

Finally, conductors or components in one circuit can act as a transmitter that radiates a wireless signal to nearby devices. Such couplings mean that practically any circuit can potentially interfere with any other close circuit or device. Keep in mind that any wire, printed circuit board (PCB) trace, or other conductor can be an antenna to send or receive radiated EMI.

Preventing or Minimizing EMI

EMI is becoming a huge problem with the growing number of electronic devices we use every day. For that reason, steps have been taken over the years to reduce EMI to acceptable levels. For instance, the Federal Communications Commission (FCC) has rules and regulations related to how much EMI must be reduced in all electronic products. Check out the extensive guidelines outlined in Parts 15 and 18 of the FCC's Code of Federal Regulations (CFR) 47 if you'd like to learn more. All new products are designed with the idea that EMI must be reduced. Here's some of the ways EMI is minimized or controlled.

Isolate Circuits or Products. EMI signals are usually weak but even so, sensitive circuits like some with high gain can still be compromised. Since coupled signals decrease in amplitude by a factor of the square or cube of the distance between circuits or equipment, the fastest and easiest way to reduce EMI is to separate the interfering units as much as possible.

When designing products, that means isolating offending generators like digital clocks from analog circuits as much as possible. Good PCB layout is critical. Keep copper connections short and spaced widely. Wiring should be short and not parallel. As for radio equipment, keep it as far away as possible from one another and from TV sets, audio devices, or anything else that may get EMI (like a cordless phone).

Cabling. One way EMI is coupled from one device to another is by interconnecting cables. If cables are run parallel to one another, you can bet that some coupling will occur. This is especially true of long cables. Transferred signals by inductive or capacitive coupling are called crosstalk. To suppress EMI, separate cables from one another as much as possible, or run them at right angles to each other.

Another solution is to use twisted pair cable that minimizes coupling and cancels common mode signals. Shielded twisted pair is even better. Coax cable is self-shielded, so is good at minimizing noise and EMI pickup. Even with this beneficial cabling, keep the cables short and prevent them from running in parallel with one another.

If giving products space does not work and cabling is not an issue, try one or more of the "big three" solutions to EMI: filtering, grounding, or shielding

Filtering. If you experience EMI you can usually get rid of it with a filter. Power line noise can be suppressed with an AC line version. This is a low pass filter that lets in only the 60 Hz from the power line and greatly attenuates all high frequency signals. These are very effective and most new products have such a filter built in. If not, you can buy external AC line filters to do the job.

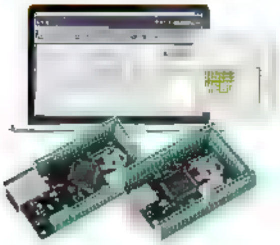
Another form of filtering is DC power bus decoupling. This is the process of bypassing the DC line with one or more capacitors to filter out power supply ripple and noise. In most cases, such decoupling capacitors are also placed near each IC power connection to suppress digital noise.

In wireless applications, all sorts of low and high pass filters are available to attenuate undesirable signals. In some cases, a notch or band pass filter may be more appropriate depending on the nature of the offending signal.

Filters at the inputs or outputs of equipment work great, but remember that they also introduce attenuation to desired signals and may cause distortion of some digital signals that may be unacceptable.

Grounding. Most electronic circuits use a ground for common connections. Poor grounds can introduce EMI. Long ground connections can be the cause of ground loops that pick up signals or introduce ground voltage drops that couple one signal to another. Large low resistance grounds are part of the solution. Keeping ground connections short is another requirement. Best of all is to run all ground connections to one common point to eliminate ground loops.

Shielding. One way to stop one signal from reaching a circuit is to use a shield. A shield is a metallic barrier



that prevents inductive or capacitive coupling. It can just be a single plane or a complete metallic box or cover, and totally block any signal transfer. As an example, digital clock oscillators are usually contained in a fully shielded enclosure or package.

Finding and Fixing EMI

If you experience EMI, you will usually know it. It manifests itself by complete equipment failure to intermittent operation. Data errors or noisy reception in a receiver are typical indicators. While you may experience some EMI effects, it is typically hard to find the source. Once you do, you can usually remedy the situation. Finding EMI is often a "cut and try" experimental process.

For example, when listening on my ham receiver recently, I was experiencing an unusually high noise level. I thought it might be the AC power line, but then in an unrelated act, I turned off a table lamp and the noise level dropped dramatically. That lamp used a CFL bulb. I turned off another CFL lamp and the noise dropped more.

If you are a ham or shortwave listener and have this problem, check for CFLs first. Plain old incandescent bulbs use more energy but do not interfere with anything.

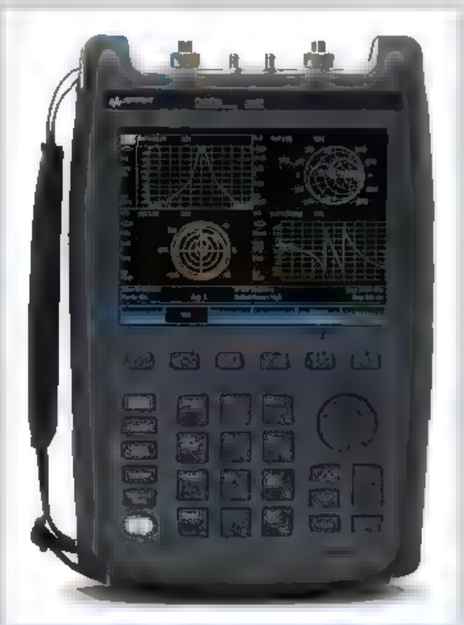
Another experimental approach is to try repositioning equipment to provide more spacing. Rearrange cables.

Motors are another EMI generator. Turn off any suspected appliances or motor-using equipment and see if the EMI goes away.

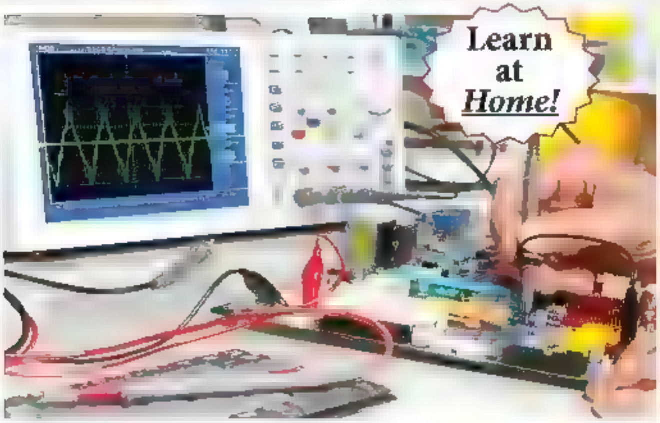
At the circuit level, you can easily find some EMI with an oscilloscope. An example is detecting power supply ripple on the DC supply line. A common offending signal is the noise from a switching power supply that has pulse waves from 100 kHz to several MHz along with their harmonics.

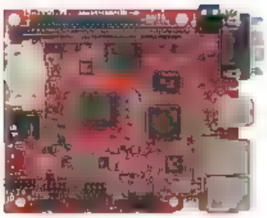
If you have a wireless EMI problem, more serious measures are usually needed. The best way to find offending RF EMI is to use a spectrum or signal analyzer. This test instrument shows a segment of frequency spectrum where you can identify your own desired signal, as well as potential interferers. Unfortunately, some of these bench instruments are often beyond the reach of us common folks. Fortunately, cheaper portable units for field use are available similar to the one in **Figure 1**. The spectrum analyzer becomes a receiver to ferret out offending signals. Some manufacturers offer an antenna or sensing probes you can use to sniff out the EMI.

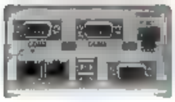
If you want to know more about EMI and RFI, I recommend the recently published book, *Radio Frequency Interference Pocket Guide* by Kenneth Wyatt and Michael Gruber. It is published by Scitech Publishing. It's a good concise summary of EMI/RFI and a helpful solution to some problems, plus it's a great book for hams having RFI issues. **NV**

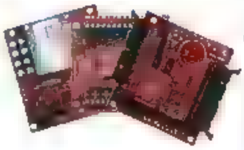


■ FIGURE 1. This portable spectrum analyzer is the best way to find EMI of the RF wireless sort. Courtesy Keysight Technologies.


Learn at *Home!*





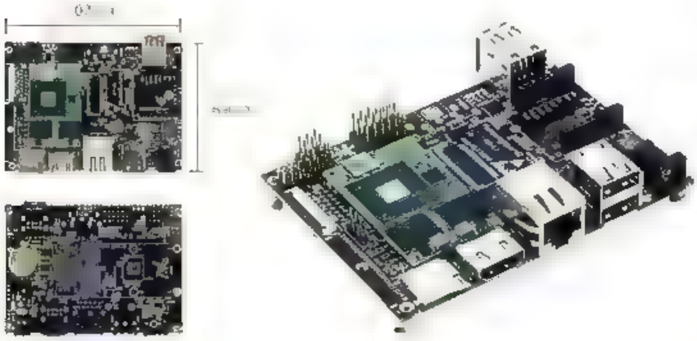


NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

ADVANCED SBC PLATFORM WITH 150+ IoT ADD-ONS

SolidRun Ltd announces the HummingBoard-Gate™ (which is based on Freescale's i.MX6 System-on-Chip) in collaboration with MikroElektronika Ltd. and the unique click board™technology.



Sized at 102 x 69 mm, the HummingBoard-Gate is the new community oriented member of SolidRun's HummingBoard family, providing a choice of processing power, memory and storage, imaging sensor connectivity, wireless/RF connectivity, and a wide range of I/Os — all powered by a variety of standard Linux distributions.

HummingBoard-Gate is the first application processor board to include an integrated native mikroBUS™socket, offering easy and simple connectivity to MikroElektronika's wide range of click add-on boards.

From simple pushbuttons to sophisticated GPS and accelerometer modules, more than 150 click boards are available, with new ones being released weekly. Professional engineers and makers alike will be able to easily expand the functionality of the HummingBoard with all kinds of sensors, transceivers, displays, encoders, motor drivers, etc. Pricing starts at $49.99.

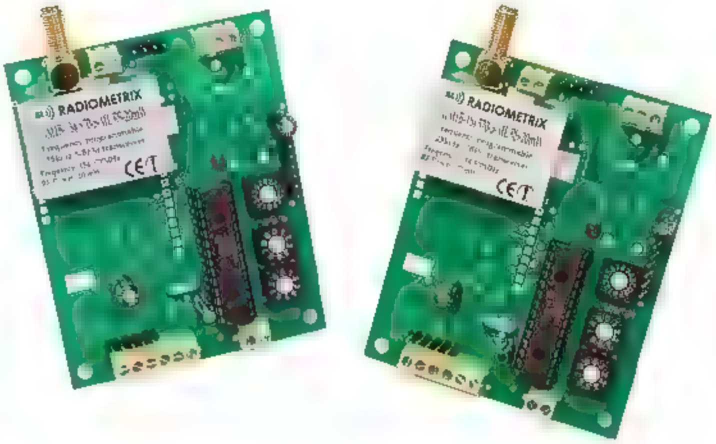
For more information, contact:
SolidRun Ltd.
MikroElektronika Ltd
www.solid-run.com • www.mikroe.com

If you have a new product that you would like us to run in our New Products section, please email a short description (300-500 words) and a photo of your product to:
newproducts@nutsvolts.com

WIRELESS RF MODULES FOR A PORTABLE ALARM SYSTEM

Lemos is now offering the A40/Radiometrix which has up to a 40 node alarm system, offering alarm outputs at each location for sounder and/or strobe lights. An optional display indicates the location of any alarm activation. With the display board, users are able to detect if a battery is low on any unit in the system, as well as a battery low indication on each node via an LED indicator or sound indicator (via an optional low battery sounder board). The whole system can be battery operated for six months on an alkaline C cell pack.

The system is a mesh network, so nodes in radio communication with other nodes allow the system to cover a large area. Because the nodes may 'hear' more than one other node, it allows for multi-path reception. The standard radios can communicate over 2-3 km in a four hop system (std 2sqkm other radios optional; country specific).

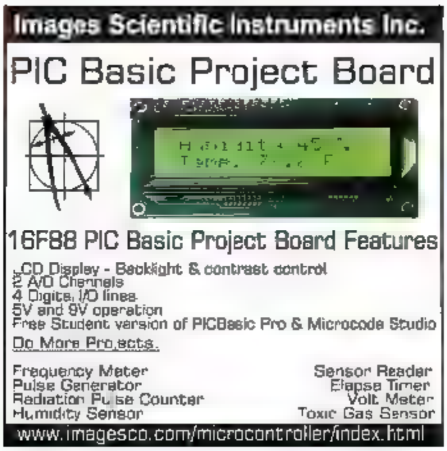


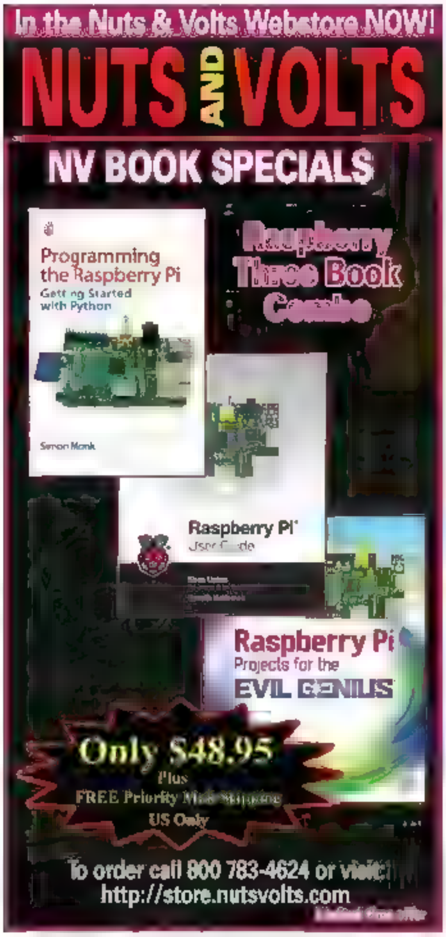
Applications include:

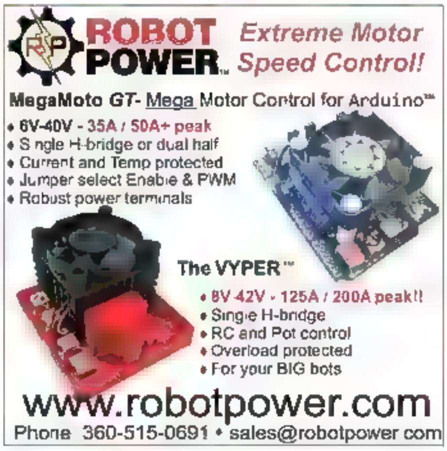
Temporary fire alarm — This is ideal for building sites or showground alarms, or anywhere a temporary alarm system is required.

Multi-building alarms — The unit can be mains powered and battery backed from an alarm system to offer a permanent solution for multi-building alarm systems.

Door alarms in buildings — Exit doors which need to be alarmed can transmit back to a central display unit with the door 'number' being displayed.







Fence alarms — With the use of gain antennas, the system could be used for border or perimeter guarding, and ranges of several kilometers could be achieved to guard areas, along with 'fill-in' units on the fence to give precise locations of intrusion.

Medical alarm in hospitals/care homes — Units could be attached to 'out of bed' alarms for elderly patients, with display units positioned on walls to show which bed has been vacated to nursing staff.

Classroom emergency alarm — Could be used in schools with a display on each node so that teachers could be alerted if a fellow teacher was being attacked. Each node would give the location of the alarm.

Light switching — Could be used to turn all lights on in a car parking area if movement was detected; a simple circuit could be used to turn off again after a certain time period. Using expanded mode, up to 400 lights could be controlled and turned on/off

Features include:

- Single input for alarm IN.
- Test mode input.
- Clear alarm input.
- Two outputs: one relay for switching up to 240V (lights, alarms); second output is a 'supply volt' switched DC output.
- Mesh network four hop system.

Continued on page 79

MONITOR YOUR CHRISTMAS TREE WATER LEVEL WITH A VOLTMETER AND AN R-2R LADDER

By Tom Bensky
tbensky@gmail.com

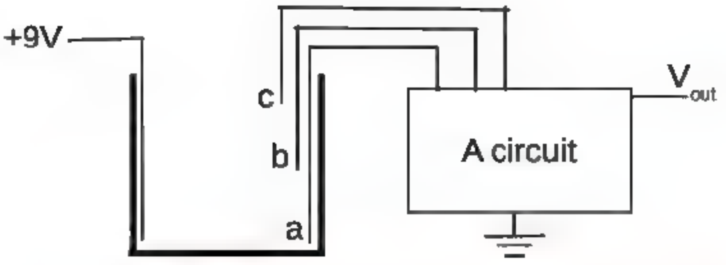
Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/november2015_Bensky.**

It's that time of year again! Christmas trees are starting to go up and with all that extra time off, you're going to need a project to work on. How about a circuit that will make it easy to check if the water reservoir in your tree stand needs to be filled or not? I thought I'd use this tree-checking problem as a reason to review voltage dividers and the R-2R ladder, and also do a bit of circuit analysis.

Let's plan on using a simple indicator of the water level: a voltmeter We'll need a circuit that outputs a voltage that is somehow related to the depth of water in the reservoir. Hmmm ... but how?

I claim it can be done by placing a positive lead at the bottom of the water reservoir, followed by three (or so) wires at incremental depths, up to the top of the reservoir as shown in **Figure 1**. Here, a 9V battery will feed the positive lead at the bottom of the reservoir. The other three wires — called "depth wires" (a, b, and c) — will be used to determine the water level. As water sits in the tank at varying depths, a charge from the 9V wire will conduct through the water into any depth wire submersed in the water (the depth wires ultimately lead to the battery's ground in the circuit). We'd like to feed all three depth wires into a circuit whose job is to present a voltage on the V_{out} wire; that will indicate the depth of the water in the reservoir.

To check the tree's water level, we'll turn on our voltmeter, push a momentary switch to turn on the circuit, check the voltmeter reading, then decide if we need to add water or not. So, what circuit can be built that can take input from three wires and output a voltage that is somehow related to the particular depth wire and be connected to 9V?

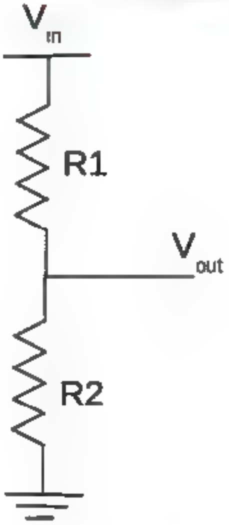


■ FIGURE 1. Schematic of proposed water-level measuring device.The"live" nine volt wire can be connected to "depth wires" (a), (b), and (c) via water of sufficient depth.

The Voltage Divider

As we figure out what the circuit should be, let's first review the voltage divider shown in **Figure 2**. t is commonly known that V_{out} will be V_{in} x R2/(R1 + R2). In other words, the resistors scale the input voltage (V_{in}) at their connection point, down by the factor of R2/(R1 + R2). V_{out} is always less than V_{in} (since R2 is always less than R1 + R2), hence the name "voltage divider." As an example, if V_{in} = 9V, R1 = 100Ω, and R2 = 500Ω, then the factor will be 500/(500 + 100) = 0.83, or V_{out} will be 0.83 x V_{in} or 7.5 volts. So, how does this actually work? What is the "secret life" of a voltage divider?

■ FIGURE 2. The classic voltage divider. V_{in} is some input voltage. V_{out} will be some fraction of V_{in}, depending on the values chosen for RI andR2.

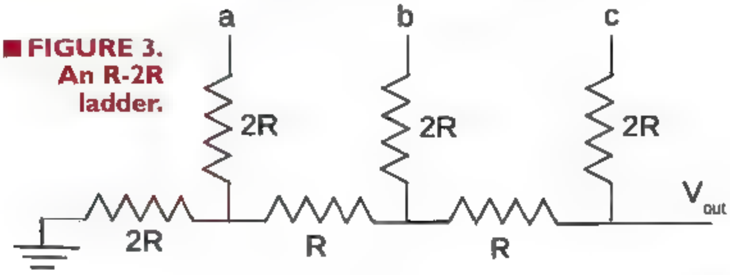


One of the stipulations of a working voltage divider is that whatever V_{out} is connected to cannot consume very much current, compared to that rushing from V_{in} through R1 and R2, and on into ground. The current between V_{in} and ground can be found from Ohm's Law, which is I = V/R. Since R1 and R2 are in series, the total resistance they present between V_{in} and ground is R1 + R2, or 600Ω (using the numbers above).

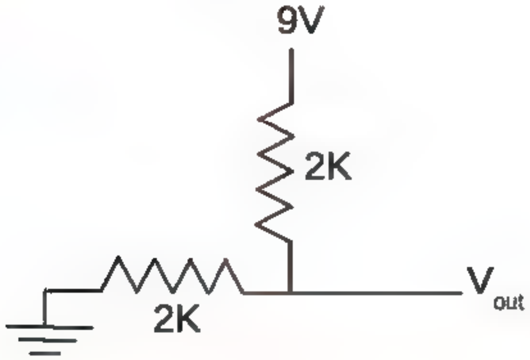
So, the current flow will be I = 9V/600Ω = 0.015 amperes (A). Thus, 0.015A leaves the battery at 9V; it then encounters R1, which presents it with 100Ω of resistance. Passing through the resistor will lower the voltage of the current by (using Ohm's Law again) V = (0.015A)(100Ω) = 1.5 volts. Thus, when the current enters R1, it will be at 9V; when it exits R1, its voltage will be lowered by 1.5V, down to 9V - 1.5V = 7.5 volts, which is the same prediction as the voltage divider equation above. So, a voltage divider works by the action of resistors lowering the voltage of current that passes through them. Keep this in mind as we go back to our tree water problem.

The Circuit

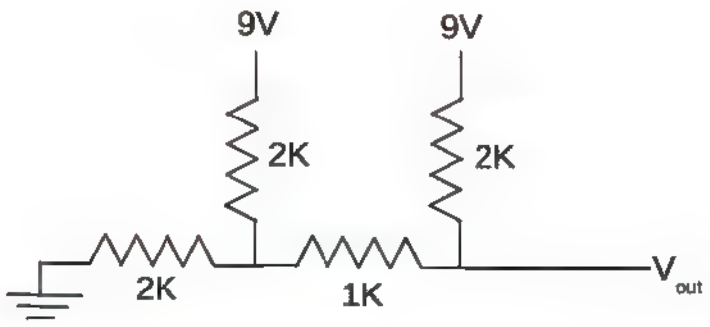
The circuit we'll route the three depth wires into is shown in **Figure 3**. Wires (a), (b), and (c) from the tree reservoir will be connected to points (a), (b), and (c) in this circuit. The circuit is called an R-2R ladder for obvious reasons: vertical resistors of resistance 2R, separated by horizontal resistors of resistance R. Motivated by our need to measure a water level and with our knowledge of voltage dividers, let's see how this circuit works. Let's use R = 1,000Ω, which makes 2R = 2,000Ω.



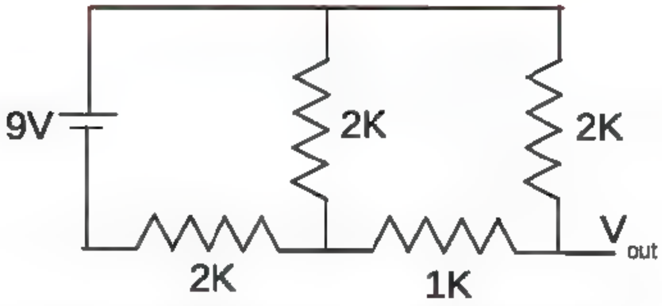
■ FIGURE 3. An R-2R ladder.



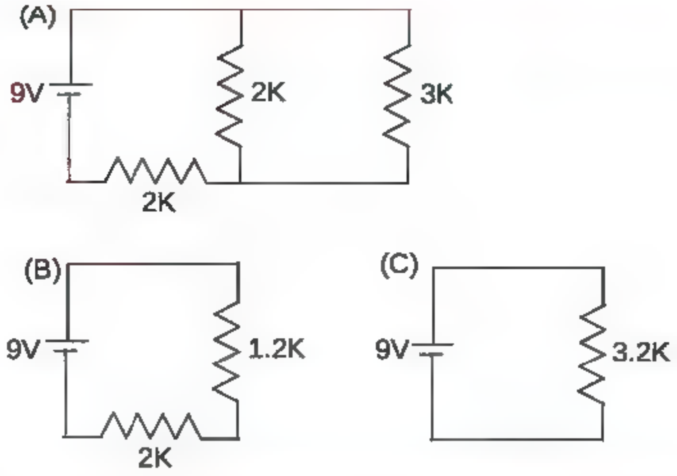
■ **FIGURE 4. The R-2R ladder from Figure 3, as it would behave if only wire (a) was connected to 9V.**



■ **FIGURE 5. The R-2R ladder from Figure 3, as it would behave if wires (a) and (b) were connected to 9V.**



■ **FIGURE 6. The circuit in Figure 5, rearranged to make analyzing it a bit easier.**



■ **FIGURE 7. (A) is Figure 6 with the rightmost 2K and 1K combined into a single 3K resistor; (B) is circuit (A) redrawn with the parallel 3K and 2K resistors combined into a single 1.2K resistor; (C) is circuit (B) redrawn with the series 1.2K and 2K resistors combined into a single 3.2K resistor.**

Suppose the water level has just covered wire (a) in **Figure 1**. In the circuit, this means (a) will be at 9V and the circuit will behave as if everything to the right of the vertical resistor connected to (a) isn't even present (since b and c aren't connected to anything). If you think at least the 1K resistors should still be in the circuit, you are sort of right, but we'll say no to this for now.

The voltmeter attached to V_{out} draws almost zero current, so virtually no current will flow through the 1Ks, and by Ohm's Law again, the voltage drop across them will be V = IR. However, with I = 0 (or nearly so), the voltage drop across either is zero. Thus, the circuit will resemble **Figure 4**.

Looking at it carefully, you'll see that it is similar to the voltage divider with equal resistors. Thus, V_{out} will read 4.5 volts, and we have our first water level to voltage mapp ng. In fact, if you read 4.5 volts, the water level is low — you might want to add some water to your reservoir. This is something about voltage dividers worth remembering: If they are built with two equal resistors, the voltage at the connection point of the resistors will be half of the supply voltage.

What if the water is covering both wires (a) and (b)? The circuit will now look like **Figure 5**. So, how can we figure V_{out} now? Well, notice in **Figure 5** that (a) and (b) are now both connected to 9V (through the water), so why not rearrange the circuit a bit? We'll tie (a) and (b) together; imagine them connected to a single battery which we'll draw toward the left, while being careful that ground is still on the left end of the 2K resistor. The result is in **Figure 6**.

Now, again assuming that the voltmeter connected to V_{out} draws no current, we see that the 2K and 1K on the right are in series. We can combine them into a sing e 3K resistor, and our circuit becomes **Figure 7 (A)**. Now, we see the 2K and 3K are in parallel, so they can be combined into a single 1.2K resistor as in **Figure 7 (B)**. Lastly, the 1.2K and 2K are in series, and can be combined into a single 3.2K resistor as in **Figure 7 (C)**.

Why go through all of this? Because the 3.2K resistor represents the total resistance the R-2R ladder presents to the battery when both w res (a) and (b) are submerged. From this, we can find the total current the 9V battery is supplying to the circuit. From **Figure 7 (C)**, it'll be (from I = V/R) I = 9V/3200Ω = 0.0028 A. Going back to **Figure 7**

(**B**), since the 3 2K (which is carrying 0.0028A) came from the 1.2K and 2K in series, both of these will also carry 0.0028A.

Now, here's where we apply our knowledge of the voltage divider from earlier. S nce 0.0028A is flowing through the 1.2K, it drops the voltage of the current by (using Ohm's Law) V = 0.0028 x 1200 = 3.38V. However, the 1.2K — with the 3.38V drop across it — came from the 2K and 3K in parallel in **Figure 7 (A)**. Thus, there's a 3.38V drop across both the 3K and the 2K.

Lastly (bear with me), since the 3K came from the 2K and 1K in series (from **Figure 6**), the 2K and 1K in **Figure 6** must have 3.38V across them as well. Finally, here's the clue to what voltage we'll get from our tree water level sensor if both wires (a) and (b) are submerged: Having 3.38V across the 2K and 1K will mean a current of I = 3.38/3000Ω = 0.0011A will flow through them. Now looking carefully, you'll see the top of the 2K resistor is at 9V and the bottom is where our voltmeter is connected. Hence, our voltmeter is measuring the drop (from 9V) in voltage across the 2K, which will be 9V - 0.0011 x 2000 = 6.74V. Thus, the voltmeter will read 6.74V when the water covers wires (a) and (b).

All told, you can see where this is going. We get 4.5V when just wire (a) is submerged. Now, we're getting 6.76V when the water is deeper and wires (a) and (b) are submerged. We won't trouble you with the calculations to figure out the voltage when (a), (b), and (c) are submerged. (Wasn't doing just (a) and (b) bad enough?) Suffice to say, it'll be a higher voltage.

While checking all of this work using Falstad's circuit simulator (**www.falstad.com/circuit**), I found that the circuit outputs 7.88V when all three depth wires are submerged. So, we have it then: a circuit that will output a voltage proportional to the water depth in our reservoir. The deeper the water, the higher the output voltage.

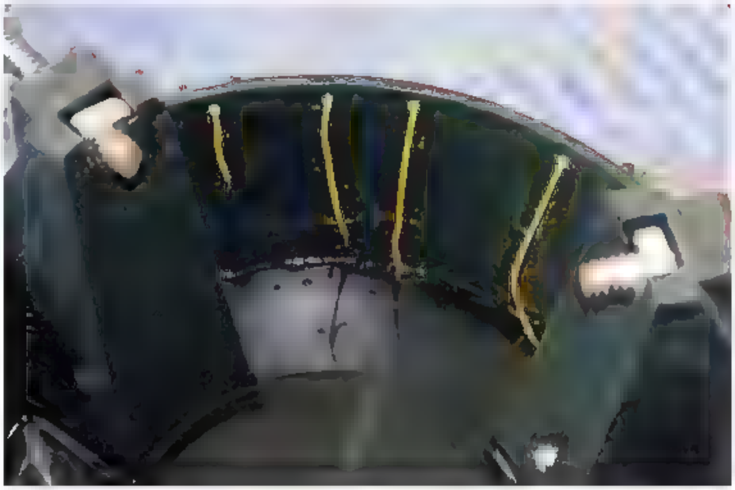
Let's build it now and see what we get.

In Practice

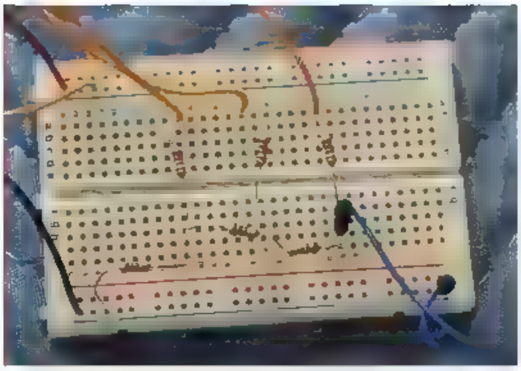
There is really nothing critical about the wiring or construction for this project. I started by wiring the tree holder as shown in **Figure 8** using 22 gauge hookup wire (I drilled some small holes near the top rim to hold the wires). In this figure, you can see the three depth wires going from left to right. My positive 9V lead is the rightmost one. Be sure to strip about 1/8" of insulation off of each wire.

Next, I breadboarded the R-2R circuit as shown in **Figure 9**. I only had 2.2K resistors in my parts bin, so my ladder is made from 1K and 2.2K resistors. I used a 9V battery clip to connect the battery to my

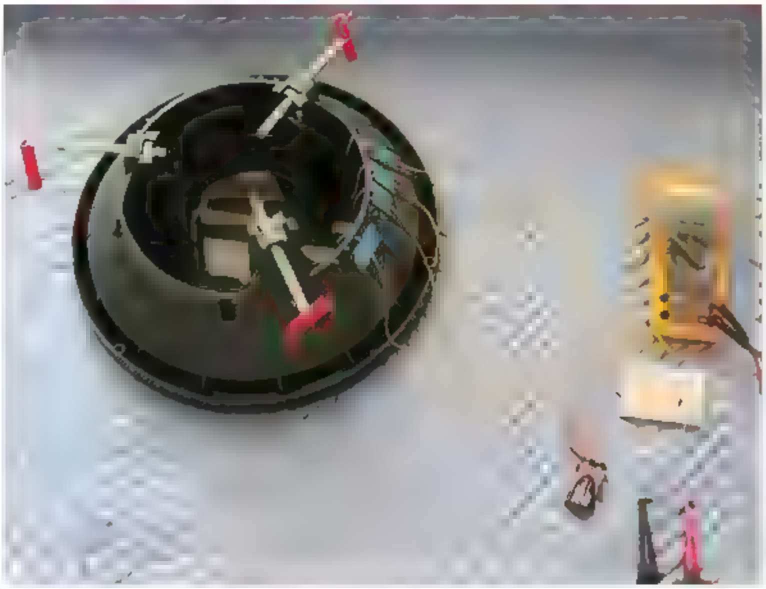
■ **FIGURE 8. Placement of the positive 9V lead (rightmost) and three depth wires in a standard tree stand.**



■ **FIGURE 9. Breadboard R-2R circuit. The yellow wire with the blue tape is the 9V positive lead. The red and black wires are from the 9V battery clip. The two blue wires are feeding the voltmeter.**



■ **FIGURE 10. Final assembly of a circuit that will output a voltage proportional to the depth of the water in the reservoir.**



ITEM	PART #	PRICE
22 gauge hookup wire	20YL-25S	$3.26 for 25'
9V battery clip	BST-81	$0.55
Breadboard	PB 400	$4.00

Parts from **Allelectronics.com**

PARTS LIST

circuit. The yellow wire with the blue tape on it is the 9V wire feeding the reservoir. The other three are the depth wires. My final assembly is shown in **Figure 10**.

When run, I get 1.8V when the lowest wire is submersed, 2.9V for the next wire, then 3.5V when all three wires are submersed. This is different than our previous calculations, but the big variable here is the resistance of the water. Probing the water quickly with my meter reads 6.3V near the hot wire and 5.7V near the other wires (both not 9V). Either way, we have an unambiguous voltage linked to the depth of the water in the reservoir.

A final version of this project would have the circuit board perhaps soldered together and mounted into a project box. Long wires would allow the voltmeter and a small pushbutton (that turns the positive lead on) to be positioned at a location that obviates the need to crawl on the floor, pushing presents aside to check the water level.

Final Thoughts

Certainly, LEDs could have been turned on by each wire as each became submersed (maybe a red, yellow, and green one, etc.), but I wanted to wrap a bit of circuit analysis into this article, which I hope you were able to wade through. The two-input case presented here would be considered a "difficult" problem to analyze for freshman science majors I teach at my university.

The idea that resistors drop the voltage of current that passes through them is a core concept in electronics, and provides a context for realizing that indeed voltage, resistance, and current (and power, energy, etc.) are all different quantities in electronics. In this case, Ohm's Law relates voltage, current, and resistance via V = I/R, and you saw two different uses of it: one to calculate a current, and the other to find the drop in voltage a resistor causes.

The astute reader will realize our original problem was really one of a digital-to-analog conversion. The digital inputs are the depth wires — either "energized" when submersed (a true state), or unconnected (false) when in the air. Our need was to map different combinations of depth wire true/false states into a unique voltage. Indeed, R-2R ladders are analog-to-digital converters.

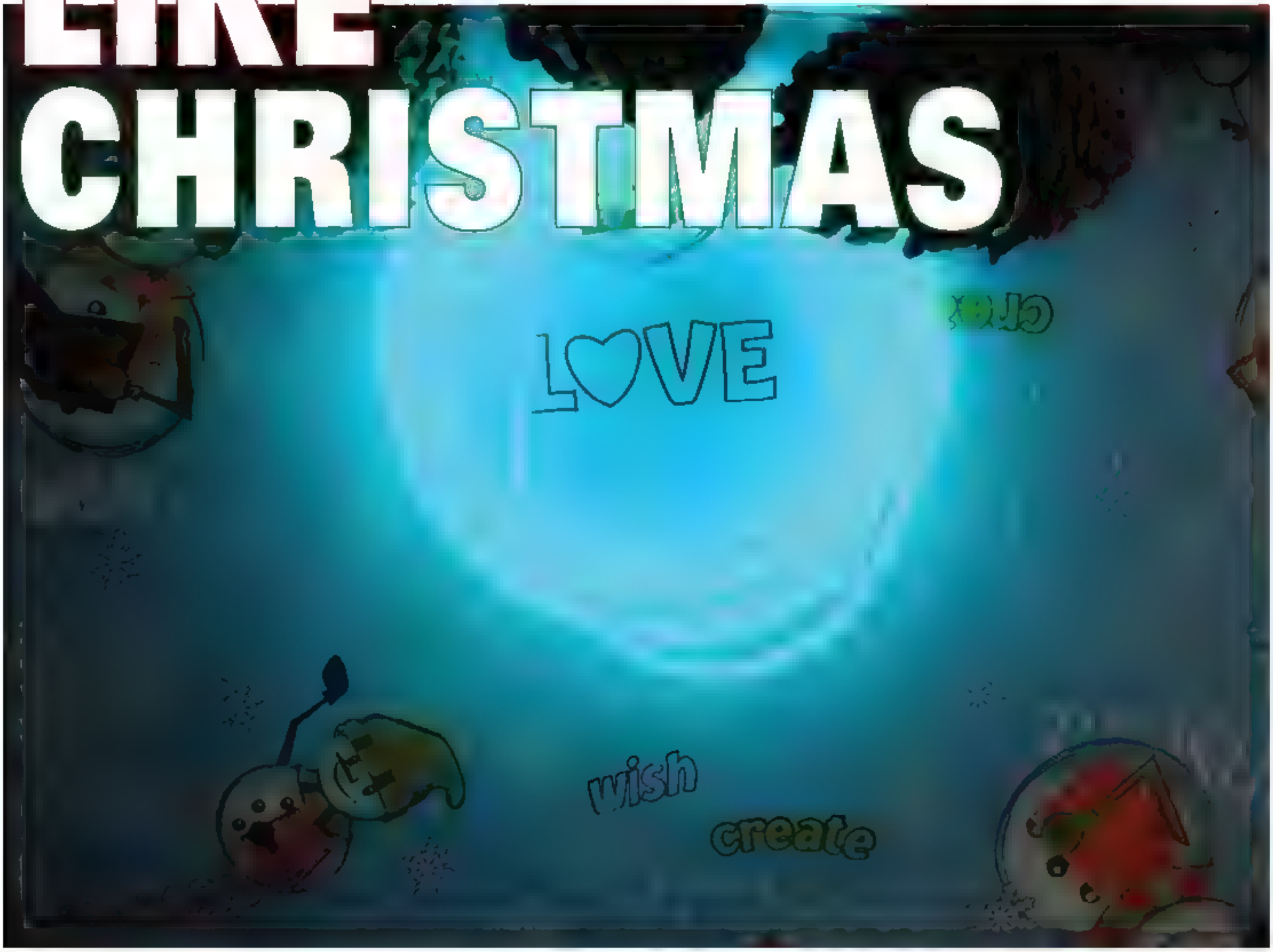
Lastly, if you need more depth resolution, you could simply add more wires and more sections of R-2R pairs. Also, having the information (depth, in this case) as a single voltage allows for many interfacing possibilities, such as sending our depth voltage into the analog input of an Arduino, for example (be careful to keep it under 5V though).

Happy holidays! **NV**

Build • Program • Learn • Compete
SERVO
MAGAZINE
FOR THE ROBOT INNOVATOR
www.servomagazine.com
Building Robots
You Can Do It!
Latest Developments
Competitions
Hacking
Programming
Sensors/Actuators
Motor Control
Design
Platforms
Microcontrollers
Electronics

By Dave Prochnow

IT'S BEGINNING TO LOOK A LITTLEBITS LIKE CHRISTMAS



Deck the halls with littleBits of projects, *fa, la, la, la, la, la, la, la, la.*

The biggest holiday season of the year is avalanching towards us. Even with only a few weeks left before the start of the season of sharing and caring, you can build a lot of great projects for enlivening all of your festive events.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/november2015_Prochnow.**

Forget the soldering iron, there's no time for that! Instead, grab your box of littleBits™(see **littlebits.cc**) and you're just about good to go. In order to refresh your memory, littleBits is the incredible ecosystem of building components that snap together with magnets and enable you to go from wish list to completed project is just a matter of minutes. In order to prove the power of littleBits, here are five incredibly fun projects that can be built in one evening – that's all five projects in one sitting, mind you! Let's get going.

Bits in Toyland

Add a glowing wreath to your front door that only lights up when the temperature drops to the overnight lows that are common in your area. This project requires the following Bits:

- Power p1
- Temperature sensor i12
- Threshold i23
- Light wire o16

Hey look at that, no other tools are required!

Figure 1 shows the four littleBits' modules used in this project connected together. The four foot long electroluminescent wire that is connected to the o16 module generates the soft blue glow for the wreath. This wire is very flexible and can be wound or braided into any diameter wreath that is selected. Use some zip ties for holding the light wire wreath in place.

After you've prepared your wreath, it's time to plug all of the modules together. In order to get the project to properly turn on at the required temperature, you will need to adjust both the temperature sensor (i12) and the threshold (i23) modules.

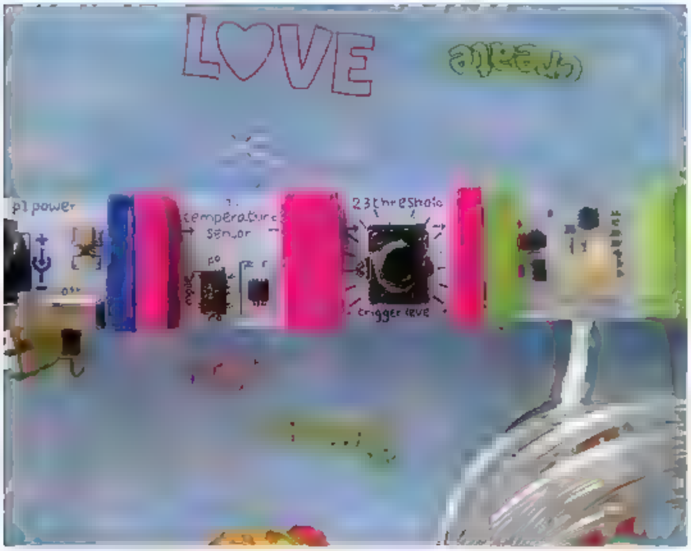
Begin by setting the temperature sensor for measuring the Centigrade scale. Next, install the entire glowing wreath assembly in its planned location. Now, monitor the temperature outdoors and when you're satisfied with the chill in the air, turn the threshold knob until the light wire just illuminates. That's it! Now, every night when the air temps drop to the programmed threshold level, the wreath will glow.

TIP: If you have a littleBits number module (o21) handy, you can temporarily remove the threshold and light wire modules and add the number module to the temperature sensor. Now as the air temps change, you can actually read the temperature on the number module. Then, you'll know exactly what temperature the wreath will glow at. Pretty nifty, eh?

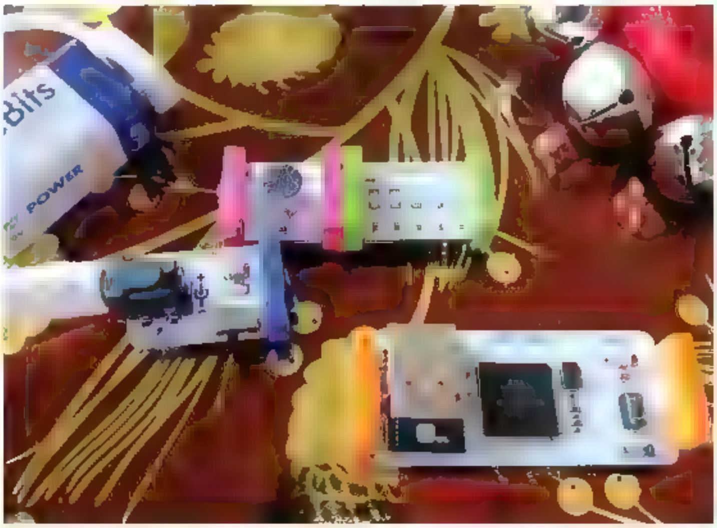
littleBits Roasting On an Open Fire

Do you have a problem with late night "guests" making trips to the Christmas tree and, shall we say, "studying" the presents? Okay, let's put it this way: Do you have kids who sneak around at night peeking at the presents? If so, then this monitor project is just for you. Here's what you'll need:

- Power p1
- Sound trigger i20
- Wire w1 (might need several of these)
- Bargraph o9 or Cloud w20

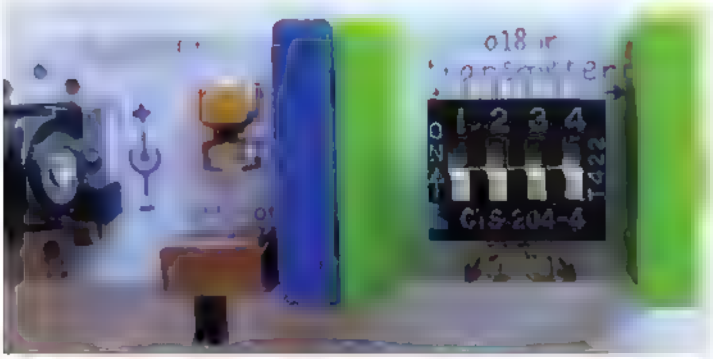


■ **FIGURE 1. This wreath is triggered by temperature.**



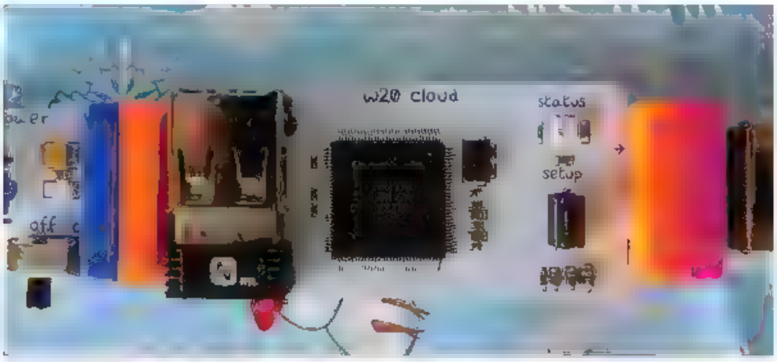
■ **FIGURE 2. Catch 'em red handed as they sneak peeks at gifts under the tree.**

■ **FIGURE 3. Santa's got your back this year.**

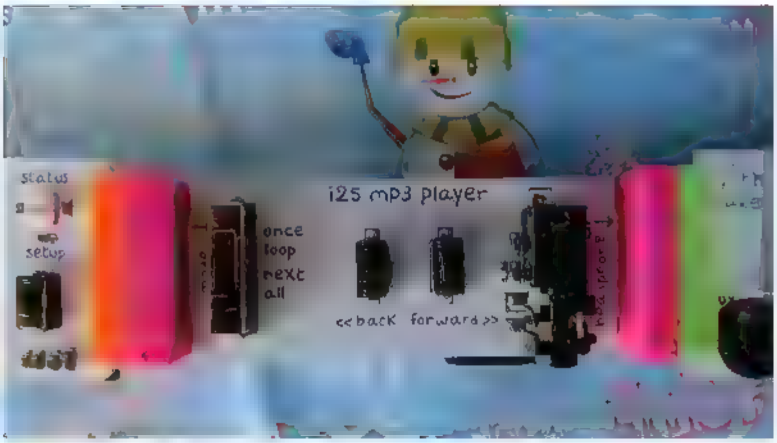


■ **FIGURE 4. Control your tree's lights with this Smart Home Kit module.**

■ **FIGURE 5. Music at your fingertips from the comfort of your smartphone.**



■ **FIGURE 6. Connect the cloud module to the Cloud Control app.**



■ **FIGURE 7. Play music with a button press inside the Cloud Control app.**

You have a choice here: Go low tech or knock their slippers off. **Figure 2** shows both of this project's choices. In the low tech route, you need to place the sound trigger module at a listening post somewhere near the Christmas tree. Then, stretch wire modules into your bedroom and plant the bargraph near your sleepy eyes. Now when "intruders" rustle near the tree, you will be flashed in the face with a five LED alarm.

A much better plan is to use the cloud as your ally. Just substitute the cloud module for the bargraph and wire modules. Once you've set up your cloud module (see **littlebits.cc/cloudstart**), it's time to head over to *If This, Then That* (see **ifttt.com**), create a free account, and make a recipe that enables the sound trigger module to send "tweets" to whomever the snoopy culprits are in your household. Oh, and make sure that your Tweets are signed by #SC.

Silver Bits

Do you make this mistake every holiday season: After adding all of the lights, decorations, and tinsel to the tree, you realize that the crazy light plug is behind the tree and woefully inaccessible? If so, this "little bit" of magic will free you from that moment of embarrassment. All you need for this face-saving miracle is:

- Power p1
- IR transmitter o18
- AC switch a3
- Button i3 (optional)

Santa's watching you this year (see **Figure 3**) and he's hoping that you plug the AC switch module in before you decorate the tree. When the last string of lights has been daisy-chained together, insert the final plug into the AC switch. Now, extend the switch module's sensor (the bug-eyed red button on the end of the switch module's cable) out away from the tree skirt so that you have a direct "line-of-sight" vision between you and the sensor.

Before you can remotely turn ON/OFF the tree, you must pair the IR transmitter module (see **Figure 4**) with the AC switch module. Just flip the transmitter's switches "ON" for a number between 1 and 4, then flash the beam towards the switch by flipping the power switch ON (see **littlebits.cc/bits/ir-transmitter**).

Once you've paired the transmitter with the switch, you can easily turn the tree ON/OFF by just toggling the power switch. On the other hand, if you want to create a more typical project, add the button module to your handheld control modules. Now with the power switched ON, you can control the tree from wherever you can see the switch sensor.

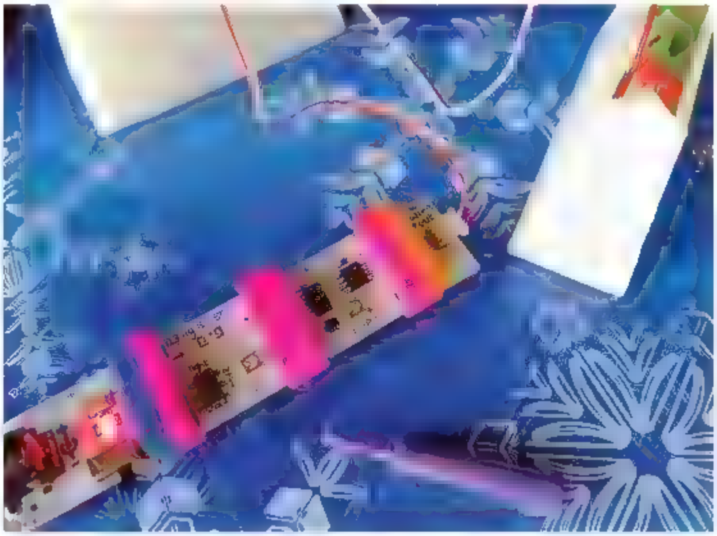
NOTE: At the time of publication, the IR transmitter module was only available with the purchase of the *Smart Home Kit*.

I'm Dreaming of a Bit Christmas

What better way to make an entrance than with Bing Crosby's "White Christmas" playing in the background? **Figure 5** shows how you can have a grand entrance. Just set up a cloud module for allowing you to remotely trigger an MP3 recording. Here are the Bits you'll need:

- Power p1
- Cloud w20
- MP3 player i25
- Synth speaker o24

Add some jolly tunes to your step this season without ever touching a Bluetooth speaker. Use the cloud module (see **Figure 6**) for pairing with the Cloud Control smartphone app. This app acts like a big remote button for triggering the MP3 player (see **Figure 7**). Naturally, you'll want to copy your preferred set of seasonal music to the microSD card inside the MP3 player. Once your playlist is ready, you can select the player to play once, loop, next, or all songs. The selected tunes are piped into the attached speaker which will triumphantly announce your arrival.

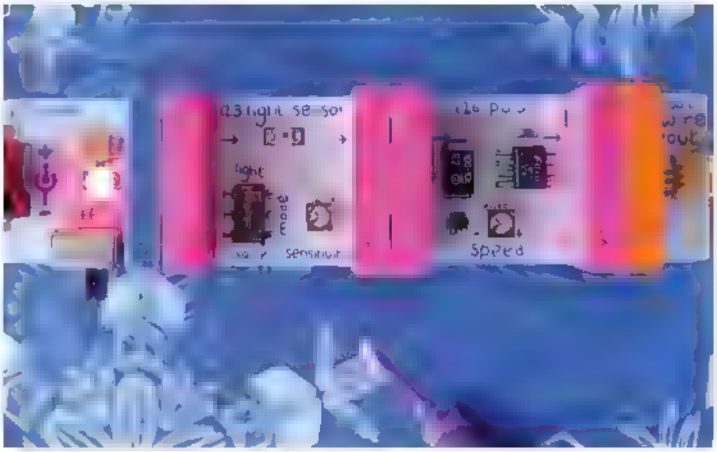


■ **FIGURE 8. Make your own light string that is controlled by ambient light.**

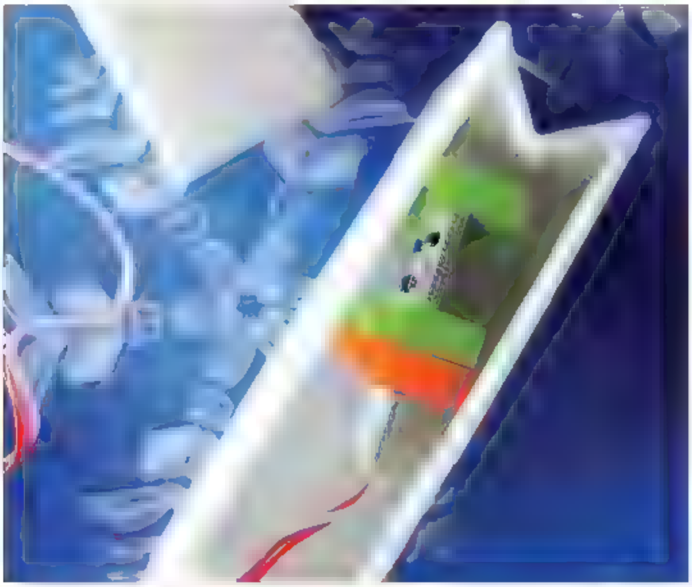
Little Drummer Bit

Finally, let's add some light to this party. In fact, let's make it so that the lights come on as the days become shorter. In **Figure 8**, you can start stringing the lights with wire and bright LED modules. This is what it'll take:

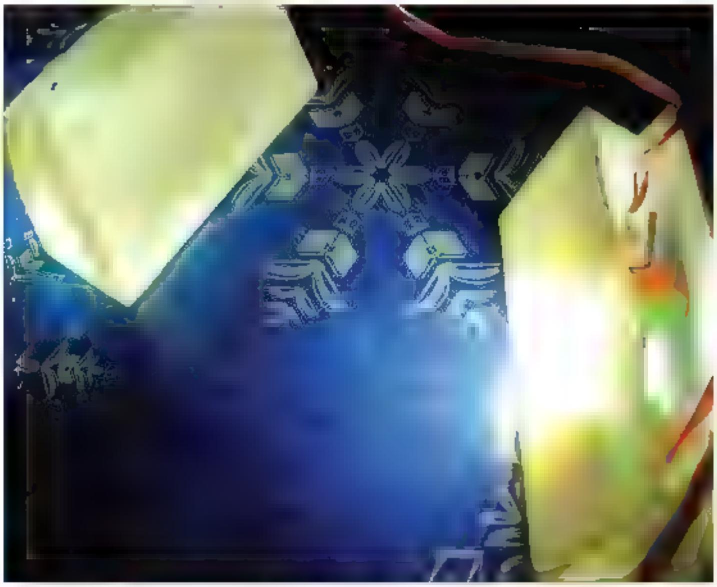
- Power p1
- Light sensor i13
- Pulse i16
- Wire w1
- Bright LED o14



■ **FIGURE 9. Adjust the light string flashing with the pulse module.**



■ **FIGURE 10. Make your own luminaries with a LulzBot mini 3D printer.**



■ **FIGURE 11. A happy holidays greeting at the front door.**

NOTE: The wire and LED modules can be daisy-chained together. You can also use split (w19) and fork (w7) modules.

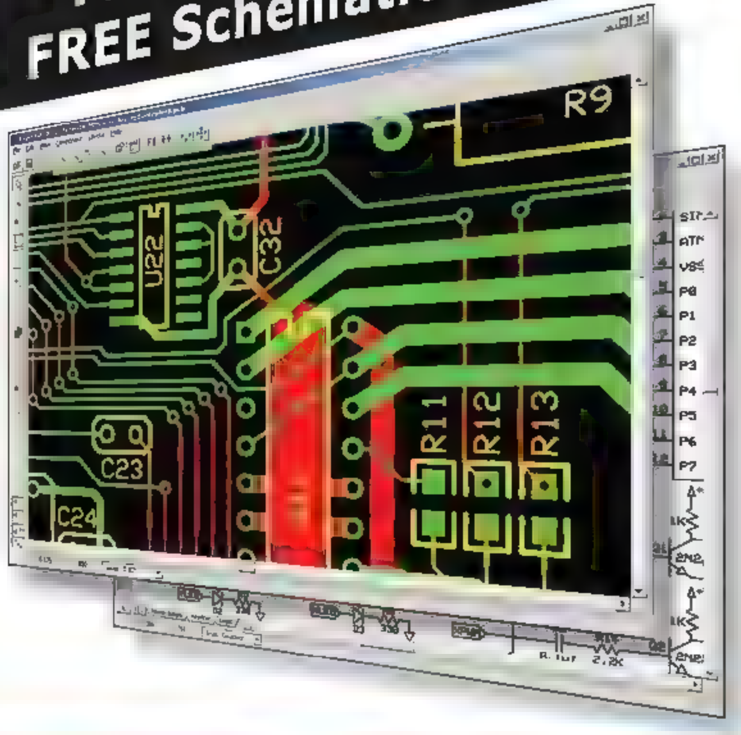
There are two variables in this project: light and pulse. The light will trigger the circuit as the room becomes darker, while the pulse controls how fast the LEDs will flash. Both of these variables can be easily controlled with the littleBits screwdriver (see **Figure 9**).

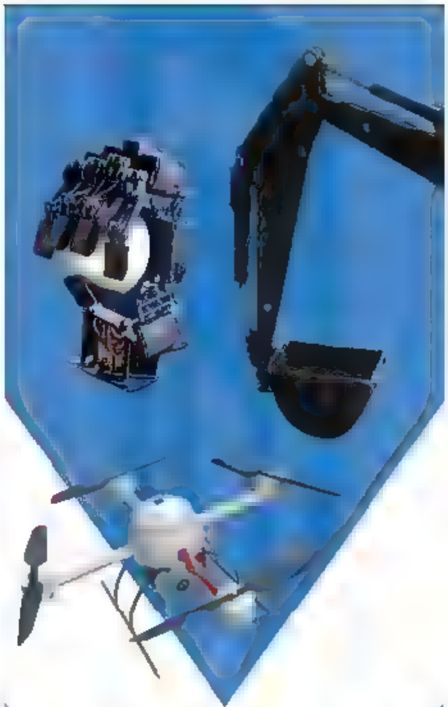
When all of the LEDs have been strung together, it's time to place them somewhere. Rather than fastening them to your mantel, how about adding them to some homebrew luminaries? The examples in **Figure 10** were made with a LulzBot mini 3D printer. Each "bag" took approximately 10 minutes to print.

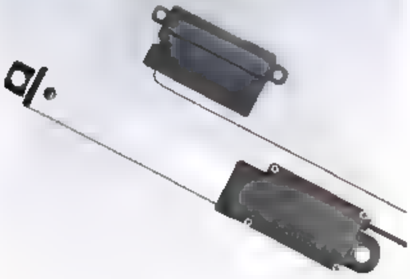
After the bags were all printed, the bright LED modules were inserted and the sensitivity of the light sensor was adjusted for triggering in low light levels. A nice and slow flash rate was set with the pulse module. The glowing luminaries in **Figure 11** now make a beautiful entrance to the holiday home.

Now, take a little bit of time and have some fun with these projects! **NV**

Second Edition
PROGRAMMING AND CUSTOMIZING THE PICAXE MICROCONTROLLER
PROGRAMMING AND CUSTOMIZING THE MULTICORE PROPELLER MICROCONTROLLER
PICAXE MICROCONTROLLER PROJECTS FOR THE EVIL GENIUS
RON HACKETT


R9
U22
C32
C23
C24
R11
R12
R13
P0
P1
P2
P3
P4
P5
P6
P7
2N2
2.2K

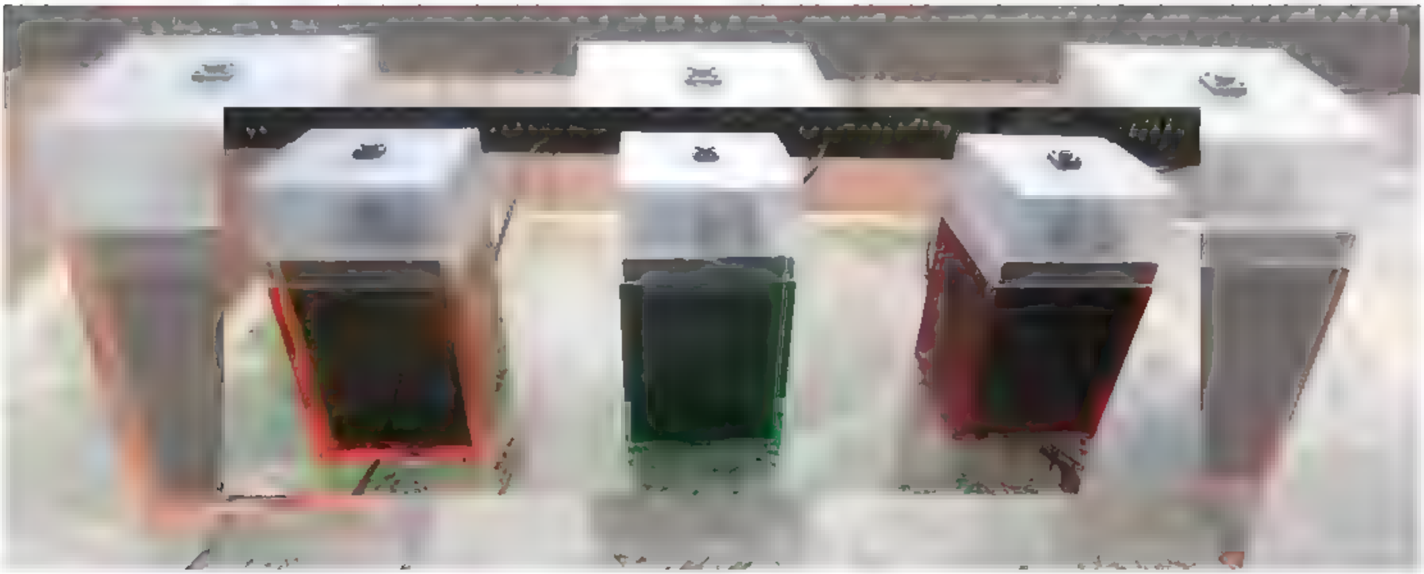




By Craig A. Lindley

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/november2015_Lindley.**

Thinking OF YOU

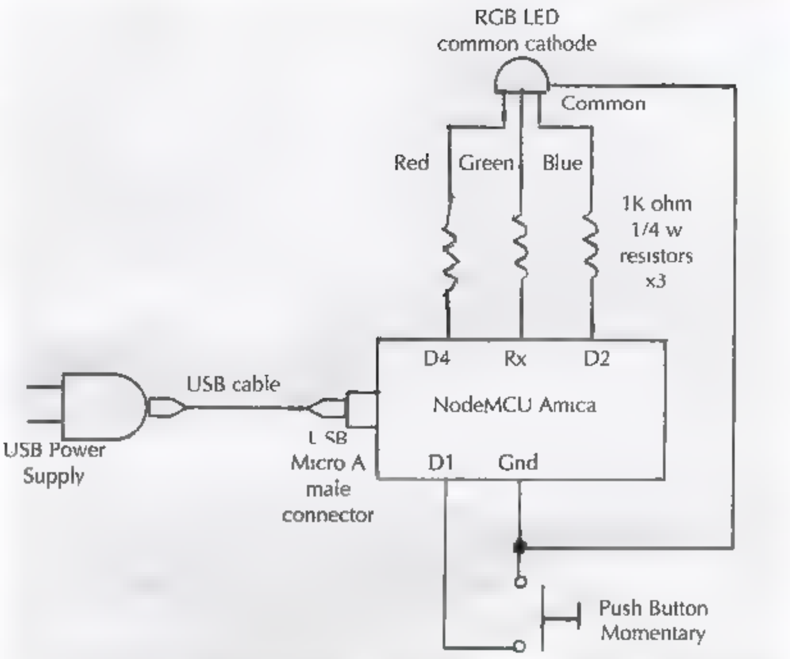


My family and friends are increasingly getting spread out across the country (or even across the globe), so keeping in touch can be a problem. This is especially true for someone like me who doesn't really like to talk on the phone much. In addition, there are problems with people's hectic lifestyles and conflicting schedules. When you think of your friend or family member, you may be too busy to call them that instant. Or, if you do, they may be too busy to talk. Thus, the spontaneity of the thought usually just fades away.

I was thinking about these issues in relation to my sisters who are located across the US. I wanted some way or thing that would virtually instantaneously let my sisters know I was thinking of them, but be totally non-intrusive. I wanted them to know without having to disrupt what they were doing with a phone call or text message, and vice versa. I also wanted them to be able to let me know they were thinking of me too.

With these thoughts n mind, I designed what I call a **T**hinking **o**f **Y**ou or ToY device.

These are small Internet connected devices meant to be used in a home, apartment, work place, or anywhere else with a fixed Internet connection. They should be placed on a desk at the office or in a room at home where they can easily be seen and interacted with. Each ToY device has a single pushbutton switch and an RGB LED for user interaction.



■ **FIGURE 1. Thinking of You device schematic.**

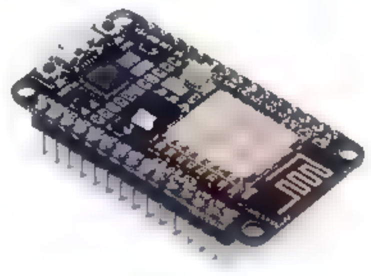
ToY devices were designed to be used in groups. For example, my family's group is made up of three ToYs: one for myself and one for each of my two sisters. Each ToY device is programmed with Wi-Fi info for each group member and assigned a colorful lighting pattern for the RGB LED that identifies the member within the group.

When these devices are powered up (they are meant to be on all of the time), they connect to the local Wi-Fi network and wait. When someone in the group presses their button (indicating they are thinking of others in the group), his/her pattern is displayed on each device no matter where they are in the world. When other members of the group press their buttons in response, the patterns are appended so all members of the group know who responded. The ToY devices continuously display the concatenated patterns for 30 minutes, then extinguish themselves until someone starts the process again.

The ToY devices are built using an inexpensive module called a NodeMCU Amica that incorporates a ESP8266-12 Wi-Fi module with an embedded application processor. I highly recommend you read my previous article, "Meet the ESP8266: A Tiny, Wi-Fi Enabled, Arduino Compatible Microcontroller" from the October 2015 issue of *Nuts & Volts* for background information on ESP8266 devices.

ToYs are really cool, easy to build little devices that I plan on giving my sisters for Christmas. (It's a good thing they don't read *Nuts & Volts* or the cat would now be out of the bag.) Maybe you should consider building these for your family, as well.

As a side note, I have some friends who live across town help me test this concept. I provided them with a ToY device and I had one as well. As we are both early risers, it always made me smile that in the early morning one of us would press the button on our device and the other would reciprocate. Our ToY devices would happily begin pulsating with our assigned patterns, letting each other know we were awake.



■ **FIGURE 2. The $10 NodeMCU Amica R2 module containing the ESP8266-12.**

The Hardware

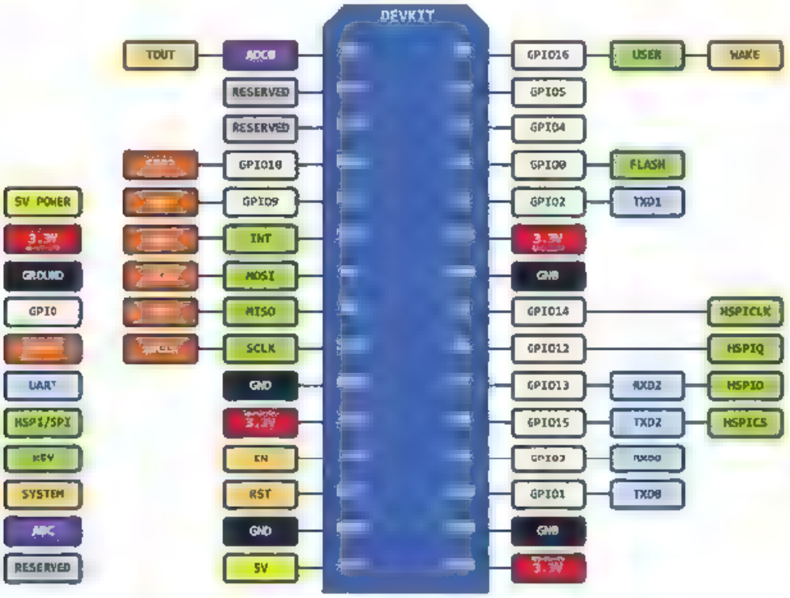
The hardware which makes up a ToY device is really quite simple and can be built by anyone with a little soldering experience in an hour or two. The components required for each unit are shown in the **Parts List** and the schematic is in **Figure 1**.

The NodeMCU Amica module in **Figure 2** is what makes this simple build possible. It consists of an ESP8266-12, along with the support circuitry required to make this module an excellent choice for Internet of Things (IoT) projects and Wi-Fi development.

The ESP8266 is a whole family of Wi-Fi modules which vary in the number of available I/O pins, the amount of onboard memory, the types of interfaces available, and in how the RF antenna is attached/implemented. The ESP8266 module on the NodeMCU Amica has its RF antenna etched directly onto the circuit board. Information on the whole family of ESP8266 devices is available at **www.esp8266.com/wiki/doku.php?id=esp8266-module-family**.

The following attributes of the ESP8266 family were extracted from its datasheet available at **nurdspace.nl/File: ESP8266_ Specifications_English.pdf**·

- 802.11 b / g / n
- Wi-Fi Direct (P2P), soft-AP
- Built-in TCP / IP protocol stack
- 802.11b mode + 19.5 dBm output power
- Built-in temperature sensor



D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

FIGURE 3. NodeMCU Amica module pinout.

- Supports antenna diversity
- Off leakage current is less than 10 µA
- Built-in low power 32-bit CPU which can double as an application processor
- SDIO 2.0, SPI, UART, and ADC
- Standby power consumption of less than 1.0 mW (DTIM3)

In other words, the ESP8266 family of modules features low power consumption, high RF power output, and is capable of supporting all of the current 802.11 standards required for Wi-Fi connectivity. In addition, it supports many industry standard hardware interfaces and can function as the application processor in many designs as it does in this one. The ESP8266 is a 3.3 VDC part.

The NodeMCU module comes from the factory with the LUA programming environment installed. However, since the ToY code was written in the Arduino environment, we will overwrite the factory programming for this application.

The ToY device uses one digital input pin for monitoring the request pushbutton switch and three digital output pins for driving a common cathode RGB LED. Current limiting for the LED is implemented using 1K ohm resistors for each of the three colored LEDs which make up the RGB LED.

To read the switch and drive the RGB LED, we must establish mapping between the general-purpose I/O pins Arduino expects and the physical pins on the NodeMCU module. **Figure 3** shows how the NodeMCU pinout helps with this task.

The mapping I have chosen for this design is shown in **Table 1**.

The ToY software configures these four I/O lines for the functions shown in the **table**

Power for the ToY module is supplied by a USB power supply/charger capable of at least one amp at five volts DC. A weak power supply will cause the hardware to operate erratically (if at all), so make sure the power supply you use is up to the task.

Table 1.

Function	Configuration	NodeMCU Module Pin	Arduino GPIO
Request switch	Digital input with pull-up	D1	GPIO 5
Red LED drive	Digital output for PWM	D4	GPIO 2
Green LED drive	Digital output for PWM	RX	GPIO 3
Blue LED drive	Digital output for PWM	D2	GPIO 4

Configuration

ToY devices must be configured before they will function as a group. For each member of the group, the following information is required:

1. A name to be associated with the ToY group member. This is only used in debugging messages available through the serial monitor.
2. The SSID or name of the Wi-Fi network the device will be associated with.
3. The password of the Wi-Fi network the device will be associated with.
4. A Teleduino key.
5. The selection of an LED lighting pattern for the device.

Each ToY device in a group must have a unique key and LED pattern. Also, each device must exist on a different Wi-Fi network. (With the current software, only one ToY device can exist on a Wi-Fi network.)

ToY devices use the Teleduino service I described in my October 2015 article to coordinate their activity. Each member of a ToY group must be assigned a Teleduino key (a string of 32 hex characters that uniquely identifies a specific ToY device) which is available for free at **www.teleduino.org/tools/request-key**. For example, if you have five members in your group, you will need five keys.

After all of the information about the group members is gathered, it must be edited into the *devices* array within the software (in the ThinkingOfYou.ino file available at the article link). An example is:

```
DEVICE devices[] = {
  “Sister1”, “S1_SSID”, “S1_PWD”, “S1_KEY”,
  redHeartPattern,
  “Sister2”, “S2_SSID”, “S2_PWD”, “S2_KEY”,
  greenBlueSweepPattern,
  “Craig”,   “MY SSID”, “MY PWD”, “MY KEY”,
  rainbowPattern,
};
```

There will be one line of data in this array for each group member.

Once the *devices* array is filled in, the Thinking of You software must be compiled and uploaded into each ToY device. The software installed for each member of a ToY group must be identical.

The Software

Although the ToY hardware is quite simple, the software is anything but. Luckily, you don't really need to know how the software works in detail to be able to use it. If you are interested in learning how the software works, read on.

The software was developed using the Arduino IDE (Integrated Development Environment). Again, see my previous article for how to set up the Arduino IDE on your computer for targeting ESP8266 type devices. Make sure to select "NodeMCU 1.0 (ESP-12E Module)" as the board type in the Tools menu.

The Thinking of You software is available at the article link; the file is called *Lindley_ThinkingOfYou.zip*. To use this software, unzip it and move the ThinkingOfYou directory into your Arduino directory. The Thinking of You software is made up of the files shown in **Table 2**.

Once you have the Arduino IDE set up correctly on your computer, have downloaded the code from the article link, and have edited the *devices* array with information about your ToY group members, you need to compile the code and upload it to each ToY device in your group. There shouldn't be any warnings or errors during the compilation and/or upload processes.

When you power-up your ToY device, you should see the RGB LED change color in sequence from green to a whitish blue color and then go off. This sequence indicates your ToY device is working correctly and that it has established a local Wi-Fi connection. If you don't see this LED color sequence, bring up the Arduino serial monitor and look for error messages which (hopefully) will lead you to the problem and subsequent solution.

Note: There is a small blue LED on the NodeMCU module that indicates the module is powered up. It is normal for this LED to be on all of the time.

How Things Work

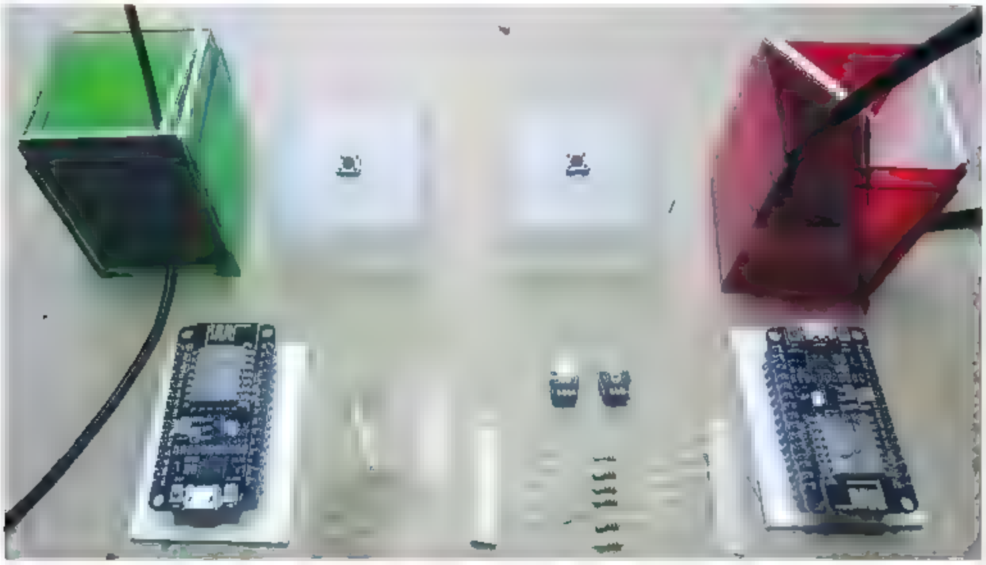
When a ToY device powers up, the network name (SSID) and password of each of the group's members (from the *devices* array) is passed on to the ESP8266. It, in turn, tries to connect to each sequentially. Once the local Wi-Fi connection is established, a persistent connection to the Teleduino server is made, and the local ToY device's key is passed to identify itself. The Teleduino server first verifies the key is valid and then begins a message exchange with the ToY device to verify it is up and running. This exchange — managed by the TeleduinoClient code — repeats approximately every five seconds to verify the ToY device is still alive and well. This process will continue as long as the ToY device is powered up.

An HTTP GET request targeting each group member is sent to the Teleduino server when any one of the ToY group members presses its button. Passed in this request is the index into the *devices* array of the device making the actual request.

The TeleduinoClient code in each group member will then receive an event from the Teleduino server which indicates which ToY device made the request. A lookup in the *devices* array then retrieves the function pointer for the LED lighting pattern corresponding to the ToY device that

Table 2.

Filename	Purpose
ThinkingOfYou.ino	The main program file. This code initializes the ToY device and then runs a rather complex state machine which polls the request switch for activity, queues requests from ToY group members for pattern display, and controls the length of time LED patterns are displayed.
LEDControl.ino	Code for controlling the RGB LED using PWM (Pulse-Width Modulation) and code for color creation and conversion.
Patterns.ino	Defines the colorful lighting patterns that can be associated with a ToY group member. Each pattern is written as a state machine. You could easily add your own patterns once you understand how this software works.
TeleduinoClient.h	The interface specification for the TeleduinoClient class. It defines a class constructor and four public class methods.
TeleduinoClient.cpp	A rather complex state machine that establishes a TCP/IP connection to the Teleduino server and interprets the data it receives back.
TeleduinoRequest.ino	Code for issuing an HTTP GET request to the Teleduino server.
Types.h	Miscellaneous data type definitions.



■ **FIGURE 4. Preparation for construction.**

made the request. This pointer is added to the LED pattern display stack. Code running in the background sees that the display stack is no longer empty, and starts the LED lighting pattern on each ToY device.

If and when a group member responds, its display pattern is appended to the display stack and the patterns display sequentially.

A background function monitors how long the patterns have been displayed. After 30 minutes of no switch activity, it clears the display stack and turns off the RGB LED.

I would like to thank my friends, Dave and Barbara Resch for helping me test the Thinking of You concept and the actual devices.

Packaging the ToY Device

I wanted the ToY devices to be as small as I could easily make them and as unobtrusive looking as possible. I definitely didn't want them to look like a bunch of wires and parts kludged together since they will need to be placed in a conspicuous place to be seen and interacted with.

Fortunately, I had just visited a Container Store and remembered these cool little vividly colored plastic boxes I thought would work perfectly. They come in all sizes, but I picked one w th dimensions of 1-5/8" x 1-5/8" x 2-7/8" which would easily fit the NodeMCU module with room to spare. I decided to buy two of these boxes for each device: a clear one and a colored one. I would use the top of the clear box and the bottom of the colored box together. This would allow the light from the RGB LED to radiate from the top of the box but would help hide the electronics in the bottom.

With that decided, I used steel wool to frost the clear top to diffuse the LED's light. I then drilled a hole in the middle of the top for the miniature pushbutton switch. I used a small file to create a square hole for the switch and then used a couple of drops of Super Glue™to hold the switch in place.

Next, I cut the small connector off the end of a USB cable as I needed to feed the cable through a hole in the bottom portion of the box. I also decided the electronics were still too visible, so I cut four pieces of black frame matting to fit inside the bottom of the box. This not only hides the electronics, but forms a support for mounting the NodeMCU module as well.

PARTS LIST

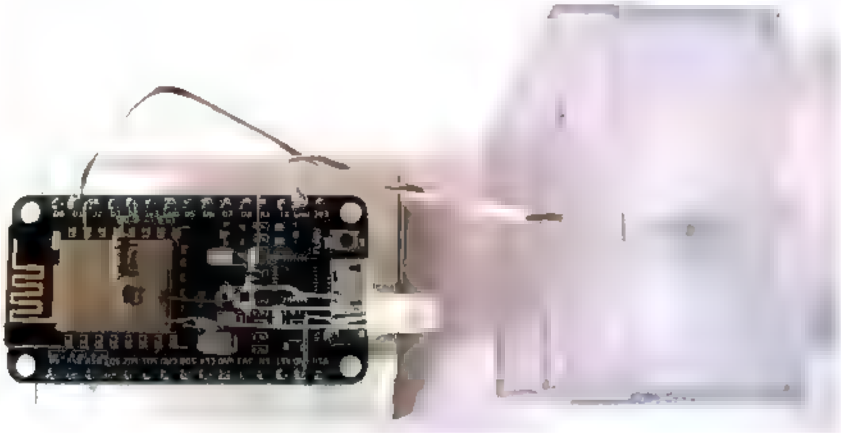
QUANTITY	DESCRIPTION	SOURCE
1	NodeMCU Amica R2 Module	electrodragon.com
1	RGB LED Common Cathode	amazon.com
3	1K ohm 1/4 watt Resistor	anywhere
1	USB Power Adapter/Charger 1 amp @ 5 VDC minimum	eBay.com
1	Micro USB Cable — A to Right Angle Micro B	amazon.com
1	Micro USB Type A Male 5-pin Connector (optional)	amazon.com
1	PCB Momentary Tactile Pushbutton Switch	amazon.com
1	Amac Box Clear #60300 You can also buy two boxes (one colored and one clear) and put the clear top on the colored body.	containerstore.com
	Frame Matte Material	Hobby Lobby/Michaels
	Super Glue	Hobby Lobby/Michaels
	Hookup Wire	Anywhere

Resources

Information about Teleduino
www.teleduino.org

Information about programming the ESP8266 in the Arduino environment
github.com/esp8266/Arduino

Information about the NodeMCU Amica
www.electrodragon.com/product/nodemcu-lua-amica-r2-esp8266-wifi-board



■ **FIGURE 5. Assembly complete. Note: The NodeMCU module is Super Glued to the frame matte material.**

Figure 4 shows two units being prepped for assembly. Here, you can see the frosted tops with attached switches, the USB cables fed through the sides of the colored boxes, the pieces of frame matting, and how each NodeMCU module was attached to a piece of the matting with drops of Super Glue™in each corner.

All of the electronic components are soldered directly to the NodeMCU module except for the switch which is wired to the module. (This can be seen in **Figure 5**.) I formed the resistor and LED wires to be as short as possible to insure they are mechanically rigid. I also made sure the LED was placed off to the side of the USB connector on the NodeMCU module so my modified USB cable could be easily attached.

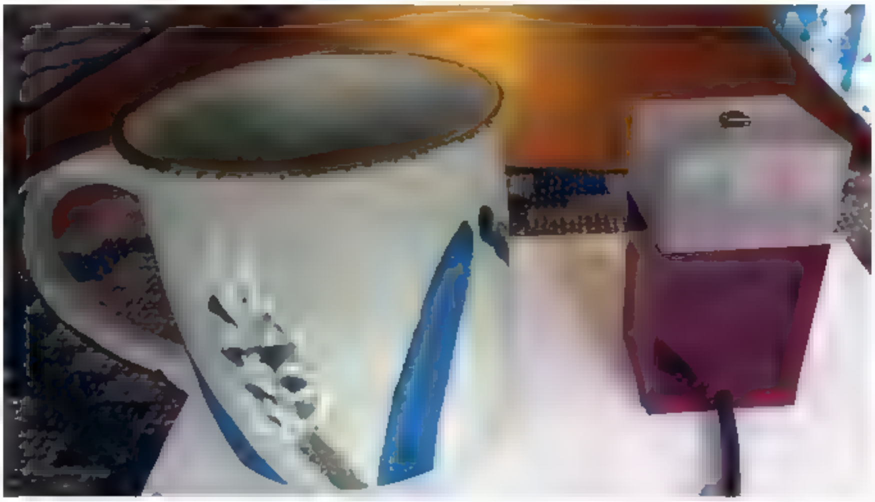
As mentioned, I had to cut the end off of the USB cable so that it could pass through a small hole in the side of the plastic box. It would be possible to strip the wires from both ends of the cable and reattach the micro USB connector, but I decided to use a new connector instead. This would allow the cable/connector to have a lower profile so it wouldn't extend up into the clear plastic box top

With the cable completed, I attached it to the NodeMCU unit and slid it into the colored box bottom. I then inserted the other three pieces of matting material and held them in place while I put a drop of Super Glue in the corners to hold them together. Placing the clear plastic box top onto the bottom completed the construction.

Conclusion

Keeping in touch with family and friends is very important in the hectic world we live in. That, however, is not the only use for a ToY device. It could be used, for example, to alert your friends across town that it is time for beer o'clock, or it could be used to monitor your child(ren)'s continued presence at someone else's house, or to inform your morning car pool members you are on your way ... the applications are endless.

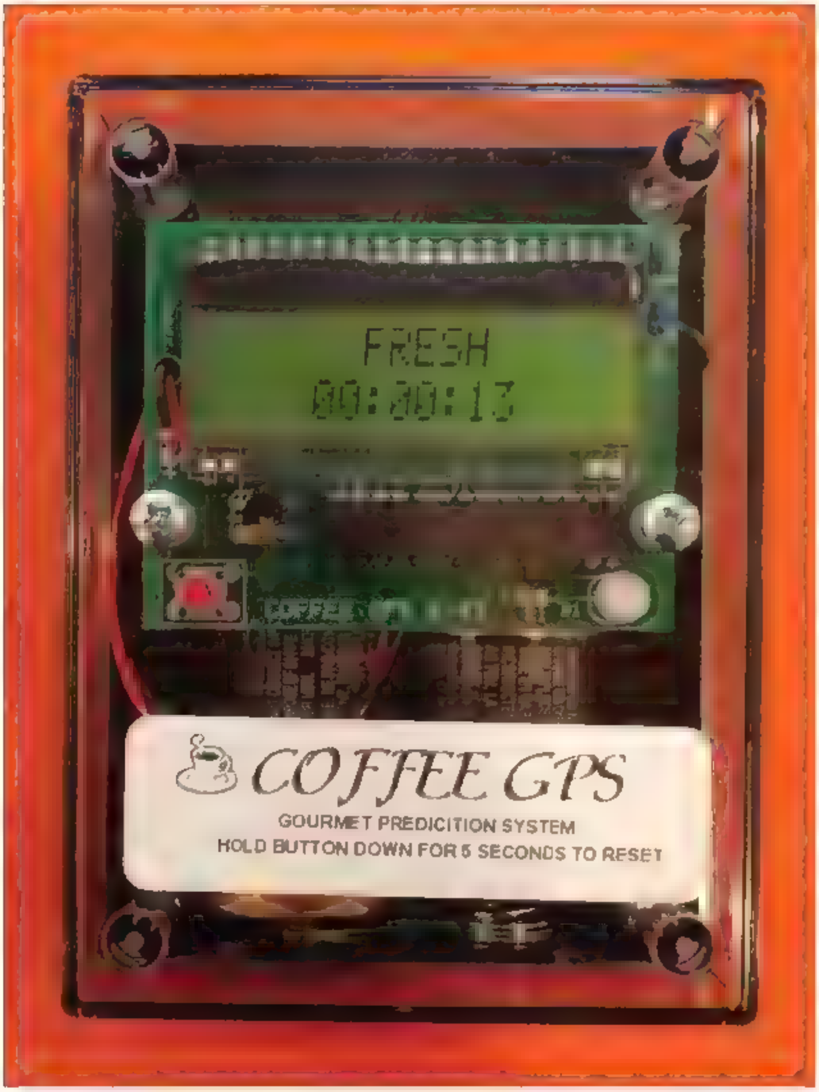
For me, I think this will make a nice Christmas present. With a touch of a button, we can let loved ones know that we are thinking of them all year long. NV



■ **ToY device is smaller than a coffee cup.**

GPS UNIT FOR TRACKING COFFEE

By Ron Newton


FRESH
00:30:13
COFFEE GPS
GOURMET PREDICITION SYSTEM
HOLD BUTTON DOWN FOR 5 SECONDS TO RESET

YUCK! This coffee tastes like mud!

When you're at work, how often do you pour yourself a cup of java only to find out it has been brewing for hours on a hot plate. In this age of electronics, we track everything nowadays. So, why not have a GPS (Gourmet Prediction System) for the taste/quality of coffee?

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com /index.php?/magazine/article/november2015_Newton.**

Coffee machines can vary widely from $10 to $10,000 trying to get many kinds of coffee just right. What is it that makes coffee bitter? Coffee has many strange compounds such as: caffeine, chlorogenic acid, caffeic acid, citric acid, malic acid, lactic acid, pyruvic acid, acetic acid, hydroxymethylfurfural, 2-methylfuran, furfuryl alcohol, trigonelline, peptides, the proteins alicyclic ketones, and aromatic ketones. Whew! (Sounds like the makings of a rap song.)

As a side note, distilled water makes coffee more bitter; soft or hard water decreases the bitterness due to the increase of dissolved solids. What does this all boil down to (pun intended)? Coffee sitting on a hot plate for a long time causes many of these compounds to combine and increase the bitterness due to oxidation/reduction potentials. So, here is a unique device that actually can have more than one application.

It is simply an elapsed timer with a few bells and whistles. There are no surface-mount components and it can be put together in a couple of hours. Three alkaline C batteries will power it for about a year. It is an ideal project for the newbie because it will teach soldering techniques and some microprocessor theory. As far as tools, a drill press is helpful but not necessary.

The Circuit

Both the board files and schematic were made using ExpressPCB's free software at **www.expresspcb.com** and are available at the article link. The assembly file for the microprocessor is also there, along with a "Hints and Tips" file which will include any updates. For those interested, a complete kit along with a pre-programmed microprocessor is available from the *Nuts & Volts* webstore. Both the LCD and the microprocessor run on 3.3-5 volts DC which makes it convenient for using three C batteries. The C batteries do add some weight to the unit, but the batteries should last over a year as the unit only draws about 0.8 mA when running and 250 µA when sleeping.

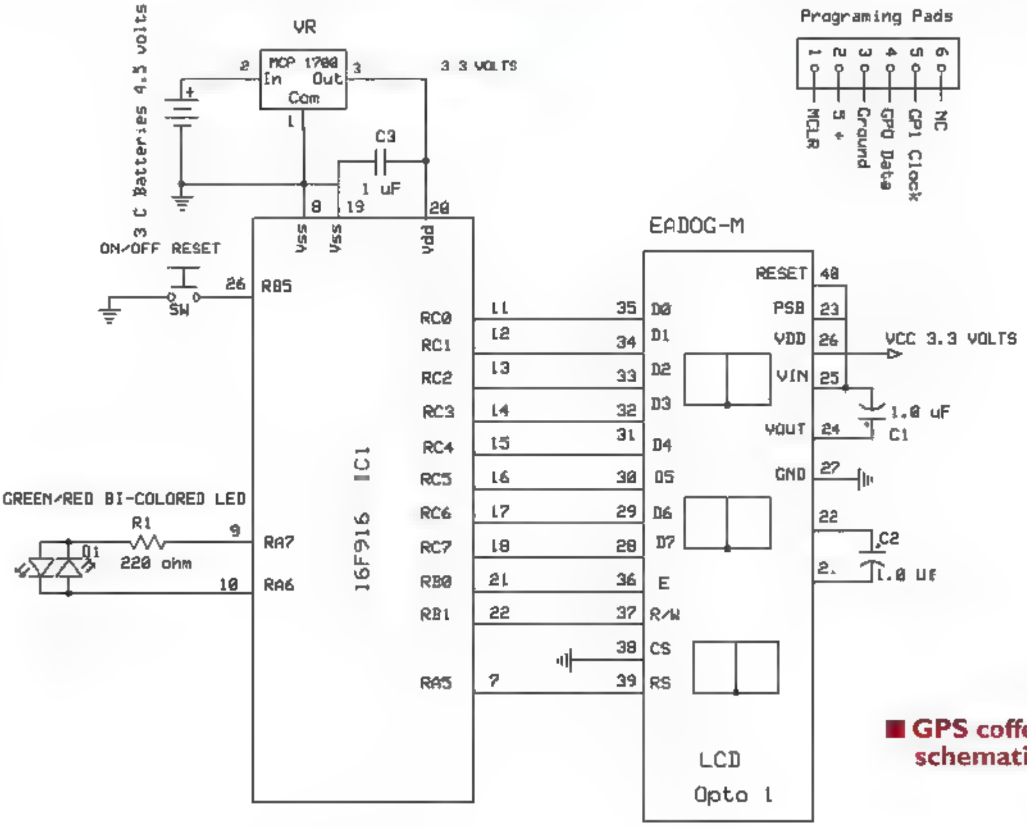
There is only one switch that controls the unit. The microcontroller determines how long the switch has been held down. After five seconds, it resets the timer. Once reset, a tap on the switch will start the timer. I also added an automatic turn-off for after eight hours. It puts the unit to sleep to draw minimum power.

The microcontroller runs at 4 MHz and (due to its design) is divided by 4, making it produce 1 MHz pulses. I used the Microchip PIC16F916's interrupt for a timing signal. Every four milliseconds, the microcontroller jumps to the interrupt vector and counts the number of seconds, minutes, and hours. I arbitrarily set the trip points for "Fresh," "Moderate," "Stale," and "Rotten" at one hour, two hours, three hours, and four hours, respectively. I also added a bi-colored red/green LED. If you'd like to use the color yellow, simply alternate between red and green pulses.

There is not a great deal of code for this project; you can view the GPS.ASM file at the article link by using Notepad. If you want to change it, I have added programming pins to the board. You can do your own programming using a PICkit 3. This will allow you to change the wording to anything you like.

LCD (Liquid Crystal Display)

Let's discuss the use of an LCD display in this project. An LCD is similar to an LED display as it is multiplexed (see



■ GPS coffee schematic.

my article on the LED Christmas tree in the December 2014 issue for a discussion of multiplexing). This project uses an Electronic Assembly Dog 3.3 version. I have used them for the last five years in a number of projects, replacing power-drawing seven-segment LED displays.

The typical power consumption is 250 µA in full operation. This allows the microprocessor to go to sleep and still display what is happening. They run on 5 or 3.3 volts with the addition of two 1 µF capacitors to generate their internal voltage. I was hoping that I could just use the 4.5 volts from the three C cells and manipulate the LCD code to run at this voltage. However, the "best laid plans" unfortunately rings true here. The LCD will not run at 4.5 volts. It either has to be at 5 volts or 3.3 volts. I ended up using a forty cent voltage regulator to maintain 3.3 volts. Why avoid using a voltage regulator? They consume power. If the unit draws 0.8 mA when running, 0.8 mA has to flow through the voltage regulator. The 4.5 volts are provided by the batteries. The regulator provides 3.3 volts output, so there is a voltage drop of 1.2 volts:

$$1.2\ \textit{volts} \times .0008\ \textit{amps} = .00096\ \textit{watts}$$

Not a great deal of power, but still a power loss which goes up in heat.

The LCDs come in three configurations: transmissive, reflective, and transreflective. The transmissive variation requires the display to be backlighted. A separate plate is added to the back that contains LEDs; they consume a fair amount of power: 3-80 mA at 3.3 volts and 2-40 mA at 5 volts. The extra backlighting display is placed under the LCD and comes in six different colors. I only use the transmissive model if the project requires reading the LCD in the dark. These LCDs cannot be wave soldered.

The transreflective configuration can be used with or without backlighting, and is black and white.

The unit used in this project is the reflective module and is yellow/green in color. It is the most readable, but cannot be backlit.

Character size varies with the number of lines. A single line display has eight characters with a height of 12 mm. A double display uses two lines of 16 characters with a height of 5.7 mm. The triple display has three lines of 16 characters with a height of 5.7 mm also. There is a function in the programming to change the size of characters. The contrast can be changed on all the different units.

The modules use a Sitronix ST7036 controller. A lookup table is used to generate 256 (2^8) characters. Eight of these characters can be of your own design. The units are talked to by four-bit or eight-bit with an SPI interface (four-wire). They can be made to blink, shift right or left, and even have a mirror image.

The LCD is initialized by calling the Dog, which will clear all the characters and initialize the module. Using the lookup table in the datasheet for the Dog 3.3, you input the hex code and call *send*. This writes the character to the LCD.

Color Changing LEDs

The bi-colored LEDs come in either 3 mm or 5 mm, and contain two LEDs but only have two leads. The diodes are put back to back. Unfortunately, I haven't been able to find a two-lead surface-mount — only four-lead. If you apply a negative voltage on the long lead and a positive on the shorter lead, the LED will turn red. Reverse the polarity and it will be green.

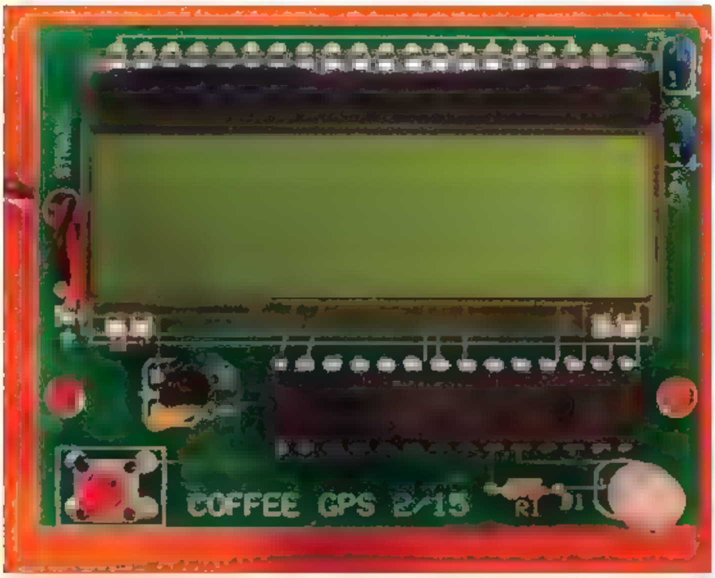
You still need a limiting resistor for this setup. The resistor in series with the LED is connected to one of the ports, and the lead from the LED onto another port. If both ports are negative, the LED will be dark. When one port going to the long (positive) LED lead is negative and the other port is positive, the LED will light red. When the port going to the long LED lead is positive and the port going to the shorter lead is negative, the LED will turn green.

However, if you alternate the ports above 60 Hz, the LED will appear yellow. This is due to persistence of vision. (This topic was also discussed in my previously mentioned December 2014 article.) By using pulse width modulation (PWM) you can change the hue, but that's a discussion for another day.

PARTS LIST

PART	DESCRIPTION	QUANTITY	DIGI-KEY #
Battery Holder	2 C cells	1	BH2CW-ND
Battery Holder	1 C cell	1	BHCL-ND
C1-C2-C3	1 µF 6 volts	3	445-8395-ND
Chassis	Serpac 4.38 x 3.25 x 2.00"	1	SR132-CB-ND
D1	Bi-colored LED T1 3/4	1	1080-1116-ND
IC1	PIC16F916	1	PIC16F916-I/SP-ND
LCD	2-line Dog 3.3	1	1481-1076-ND
R1	470 ohm 1/6 watt	1	470EBK-ND
Spacer	3/8" #6	2	492-1106-ND
VR	3.3 volt TO-92 Microchip	1	MCP1700-3302E/TO-ND
Wire	Black 24 stranded 8'	1	
Wire	Red 24 stranded 6"	1	
Screws	5/8" 6-32	2	
Batteries	C	3	

So, why flash the colored LEDs? At first, I was just going to have them on all the time until I calculated the power draw on the three C cells: Running at 3.3 volts for 1 hour / 470 ohm ≅ 7 ma/ hour. This gives a battery life of 35 days. However, if you pulse the LED for 0.1 seconds every three seconds, this extends the life of the battery for over a



■ **FIGURE 1. Coffee GPS board.**

year. There are 3,600 seconds per hour:

$$3 \text{ second interval} = \frac{1200 \text{ seconds x } 7 \text{ mA}}{3600 \text{ seconds/hr}} = 2.3 \text{ mA hrs}$$

*However, on for only .1 seconds = 5 mA hrs * 0.1 = .23 mA hrs*

A negligible amount of power goes to an alkaline C cell which has about 6,000–8,000 mA hours. However, the flash still catches the eye.

Building the GPS

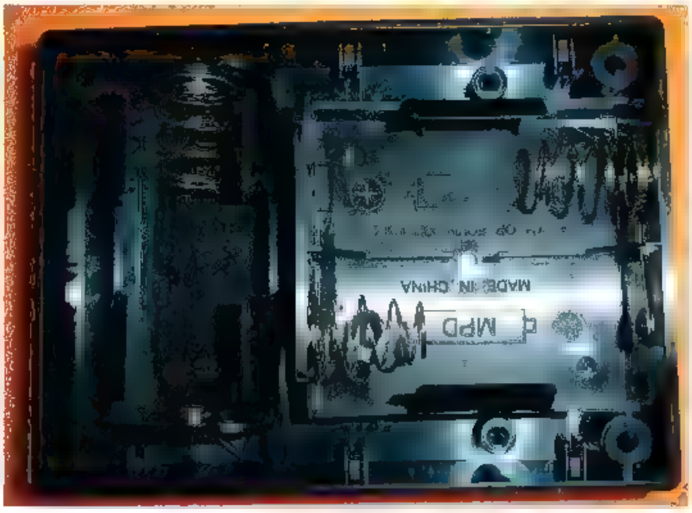
Solder the PIC16F916, noting pin 1 goes to the square pad. Solder the LCD and the switch. Solder the three 1.0 µF capacitors. These do not have polarity due to their voltages and construction. Solder the voltage regulator, noting its flat rotation to the silkscreen.

Pass the battery holder wires through the strain hole; solder the red wire to the positive pad and the black to the negative pad. Cut the black wire to the same length as the red one. The remaining wire will be used to connect the battery holders.

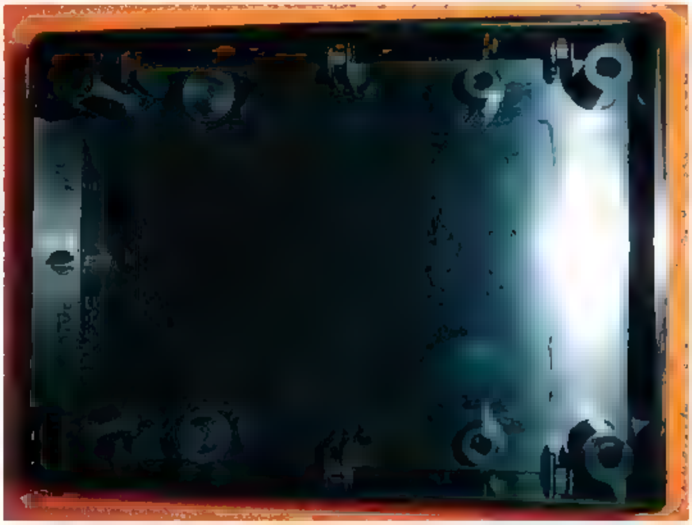
Using one of the 3/8" spacers, place it next to the LED to establish its height. Solder the bi-colored LED to the board with the long lead going to the square hole. Refer to **Figure 1**.

Go to the article link and download the GPS template. Cut out the template and glue it to the top of the box using a glue stick. Drill the four holes as indicated. Use hot water to remove the template. Place the two-battery holder into the bottom of the box. Remove the two posts either using a mill or a Moto tool for placement of the single C holder. Refer to **Figure 2**.

Solder a short piece of wire from the positive terminal of the single C holder to the negative terminal of the double C holder. Solder the black wire coming from the board to the negative terminal of the single holder and the red wire to the positive terminal of the double holder. Use double-sided tape to attach the holders to the bottom of the box. Refer to **Figure 3**.



■ **FIGURE 2. C holder placement.**



■ **FIGURE 3. Milled out posts.**

Put two 5/8" 6-32 screws through the top of the box and turn it over. Add two 3/8" spacers to the screws. Attach the screws into the board; the board should self-tap.

Using the GPS

Add three alkaline C batteries, noting their polarity. Push the switch. The timer will count upward with hours, minutes, and seconds. The LED should blink green. To reset the unit, hold down the button until the display changes.

I love coffee, I love tea, I love *Nuts & Volts* as it is good for me. **NV**

Universal Oscillator

Topologies and Applications

The secret to getting the most bang for your buck and performance is using the proper circuit topology for your particular application.

By Ron Hoffman

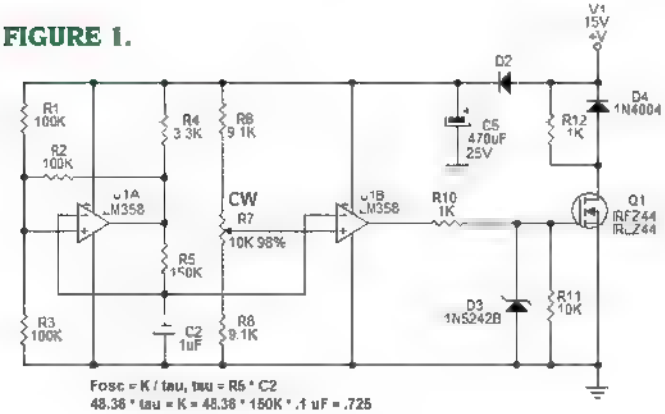
There are many types of oscillators used for many different applications. From motor speed control to light dimming, to switching power supplies, to RF broadcasting, to fiber optic data transmission, and on and on. The 555 timer IC is a low cost circuit used to generate PWM (pulse width modulation) clock pulses when used in its astable oscillator mode. It is used for car turn signals, flashing lights, and a myriad of other applications.

There is one problem with the 555 that to me renders the part useless in PWM applications. It can never always be 100% off or on. At best, its range is about 1% to 99%. So, a motor or light can never be completely off or on. That is terrible!

There is a much better solution in an eight-pin package that provides maximum on and off control, PWM modulation, FM modulation, and many more features. What is this magical IC, you ask? It is a dual op-amp chip such as the LM358 or MCP6022.

We'll explore and analyze several circuits and their applications in this article to show how effective and easy it is to design and use this circuit topology. Let's begin by looking at the circuit in **Figure 1**.

FIGURE 1.



Fosc = K / tau, tau = R5 * C2
48.36 * tau = K = 48.36 * 150K * .1 uF = .725

The heart of the oscillator is U1A, R1-R5, and C2. R1 and R3 divide the V1 supply voltage in half, and their combined resistance is R1*R3/(R1 + R3) = 50K. Since R2 = 100K, when the output of U1A switches from ground to 15 volts (+V1), the junction of U1A+ and R1-R2-R3 switches from five volts to 10 volts. R5 starts charging C2. When C2 charges above 10 volts, the input of U1A- is higher than U1A+. That causes the output of U1A to switch to zero volts.

In turn, the junction of U1A+ and R1-R2-R3 switches from 10 volts to five volts. R5 begins to discharge C2. When C2 discharges below five volts, the input of U1A- is lower than U1A+. The output of U1A switches from zero volts to 15 volts (+V1), and the junction of U1A+ and R1-R2-R3 switches from five volts to 10 volts. R5 begins to charge C2 and the cycle keeps repeating. C2 is constantly ramping up and down between +V1*1/3 (5V) and +V1*2/3 (10V).

The really cool thing is that this circuit oscillates at the same frequency even if the supply voltage +V1 changes! That is one of the big features of the 555. U1B acts as a comparator to provide the 0% to 100% PWM output. R6, R7, and R8 form a resistive divider. Notice that R6 and R8 are 10% lower resistance than potentiometer R7. Doing the math, 15*(R7 + R8)/(R6 + R7 + R8) = 15*19.1K/(28.2K) = 10.16 volts at full clockwise (100% on), and 15*R8/(R6 + R7 + R8) = 15*9.1/28.2 = 4.84 volts at full counter-clockwise (0% off).

The U1B- input is connected to the junction of R5 and C2 which ramps up and down between five and 10 volts. U1B+ is connected to the wiper on potentiometer R7.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/november2015_Hoffman.**

Whenever the ramp voltage is higher than the wiper reference voltage, the output of U1B goes high; whenever the ramp voltage is lower than the wiper reference voltage, the output of U1B is low. Since the wiper reference voltage can be set higher than the highest ramp voltage, 100% on time is possible.

Additionally, since the wiper reference voltage can be set lower than the lowest ramp voltage, 0% on time (100% off) is possible. This cannot be done with a 555 timer. So, off can really be off, and on can be full on.

The output of U1B is connected to a switch to power the load. In **Figure 1**, the output is connected via R10 to the gate of Q1 which is an IRLZ44 MOSFET transistor. D3 and R11 protect the gate of Q1 from being over-driven with voltage. R12 provides a light resistive load to Q1, while D4 protects Q1 from inductive load voltage spikes. An alternate design is shown in **Figure 2** with a current mirror-limit formed with R12, R13, R14, and transistors Q2, Q3, and Q4. With R12 and R14 equal to 20 ohms, the current to the LED strings composed of D4-6 and D7-9 is .6V/20 or 30 mA.

The two strings of LEDs use about 60 mA total. Resistor R10 feeds Q1 that is used as a bipolar switch. With the oscillator frequency set at 322 Hz, there is no visible flicker and the duty cycle of the PWM output effectively dims the LED's brightness.

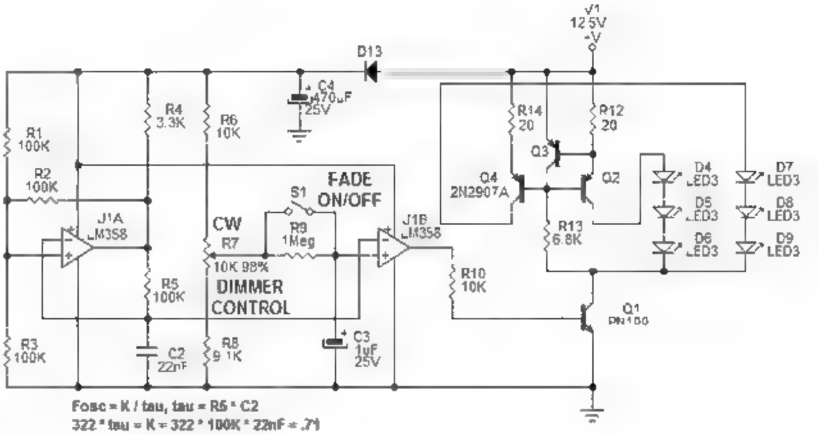


FIGURE 2. LED light dimmer application with on/off fade.

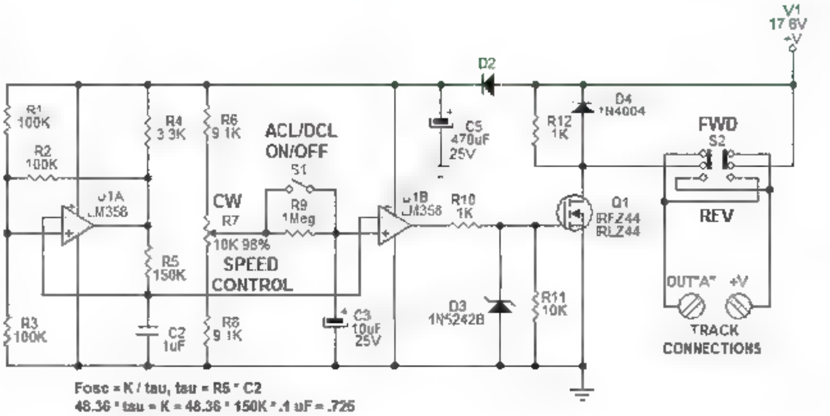


FIGURE 3. Motor speed control application with ACL/DCL.

Notice too, that S1, R9, and C3 have been added to the circuit. When S1 is open, the wiper reference voltage changes very slowly as R7 is adjusted up or down, providing a slow fading effect. If used with a motor — such as on a model railroad train — the effect is slow acceleration or a momentum effect. Such a circuit is shown in **Figure 3**.

Switch S2 provides the forward/reverse polarity function. The ACL/DCL time constant (set by R9 and C3) is around 30 seconds. That is the time needed to accelerate to full speed or decelerate to a stop.

Figure 4 shows an IR transmitter application. The PWM is modulated by U1C. As the audio input varies, U1C acts as an inverting buffer, centered at V1/2 (2.5V). The changing input voltage causes the reference voltage at U1B+ to change. U1B is varying from V1/3 to V1*2/3. The output of U1B therefore varies in duty cycle from 0% to 100%, depending on the amplitude of the audio input.

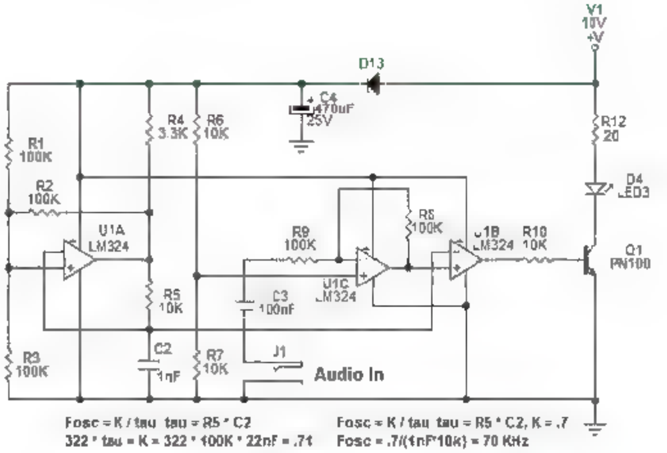


FIGURE 4. PWM modulated oscillator circuit.

FIGURE 4A. Modulated PWM waveform.

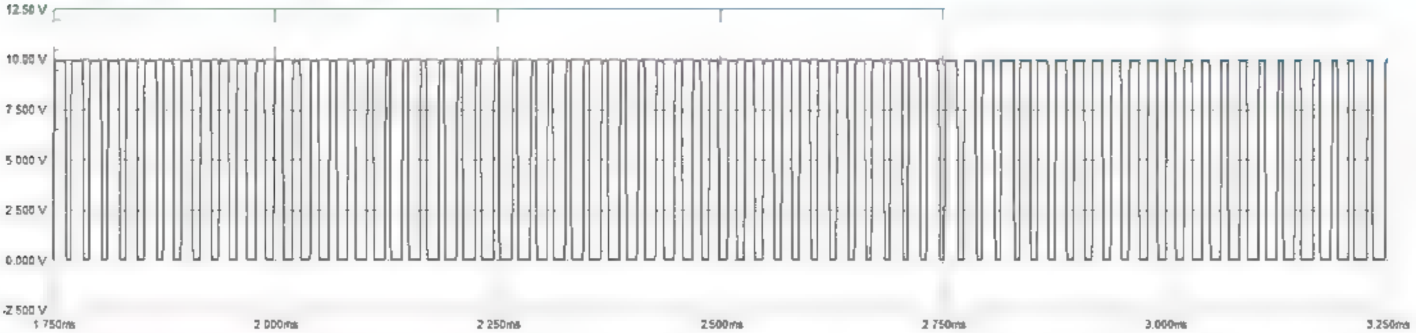
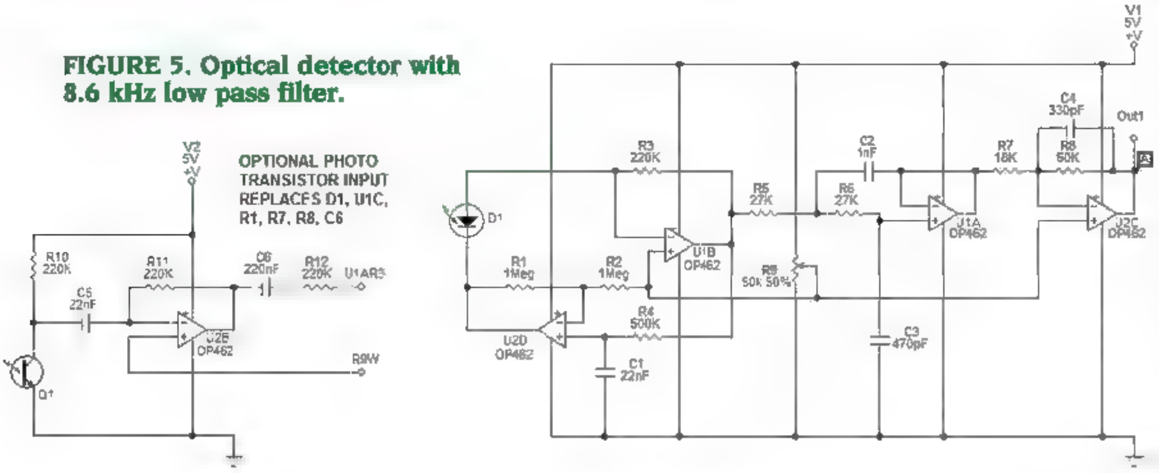


FIGURE 5. Optical detector with 8.6 kHz low pass filter.



By making the frequency very high (50 kHz to 100 kHz), the PWM beam of light can carry the audio input signal information. This works like AM radio modulation. The output wave form is shown in **Figure 4A**. Notice how the on and off times vary as the audio varies.

A receiver (refer to **Figure 5**) that detects the IR beam can recover the information by amplifying the signal and processing it using a low pass filter (integrator) — much like an AM radio signal is recovered.

The on time of the light pulse is similar to the amplitude of the RF envelope of an AM radio signal.

By modifying the oscillator circuit as shown in **Figure 6**, it is possible to FM (frequency modulate) the IR oscillator. The resulting modulation and output waveforms are shown in **Figure 6A** and **6B**. The circuit operation can best be

FIGURE 6. FM modulated oscillator circuit with IR output.

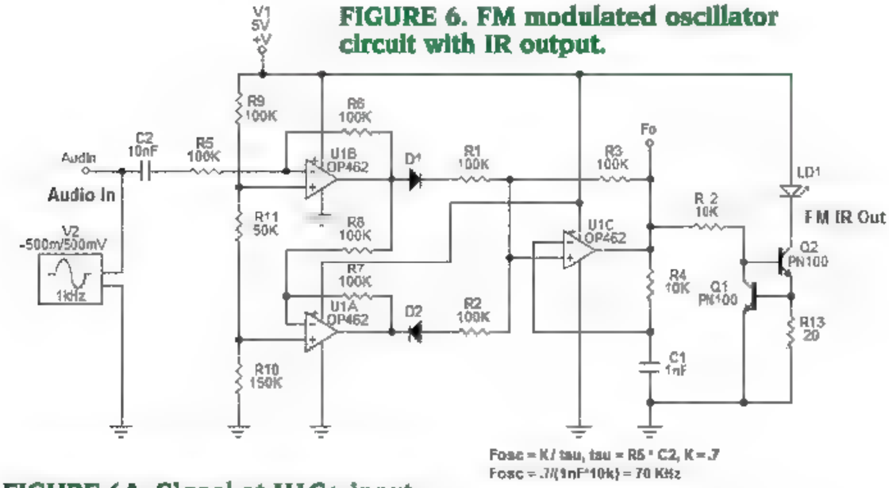


FIGURE 6A. Signal at U1C+ input.

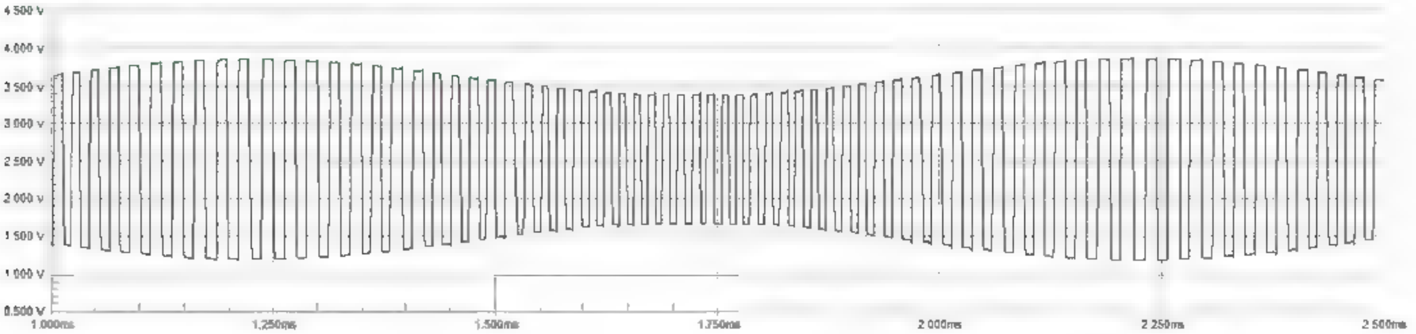
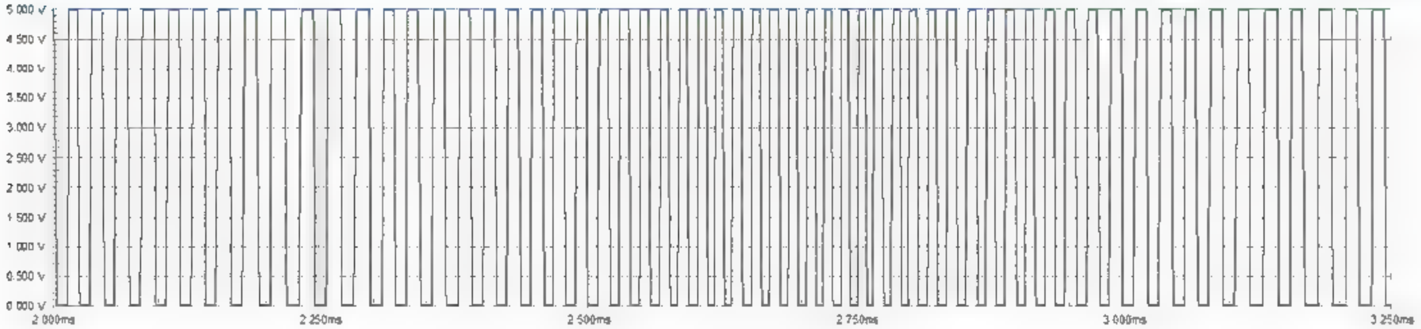


FIGURE 6B. Signal at U1C output.



described as follows.

Resistor R9, R10, and R11 form a 2Vs/3 and Vs/2 divider reference to U1A and U1B. Therefore, the output of U1B is 2Vs/3. Since U1A references the Vs/2 point, has a gain of -1, and is connected to the output of U1B (U1B is Vs/6 above Vs/2), the output of U1A is Vs/3.

When a signal is fed through C2 to U1B and the audio gets more positive, the output of U1B decreases and the output of U1A increases.

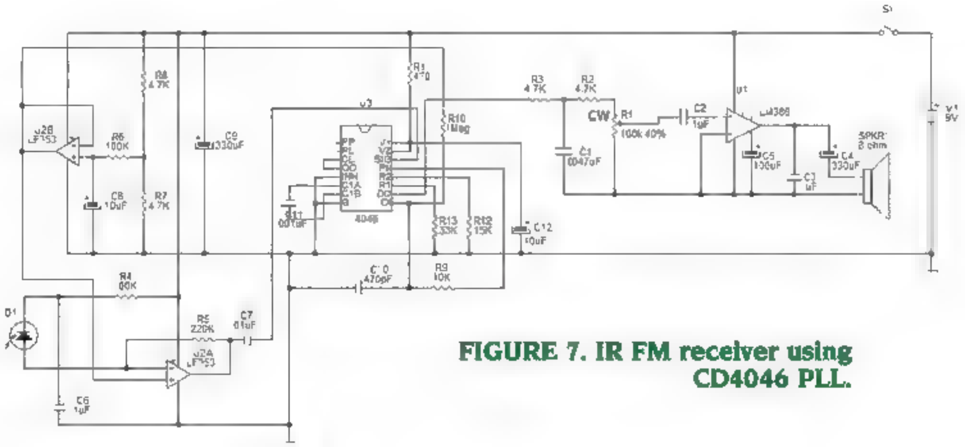


FIGURE 7. IR FM receiver using CD4046 PLL.

Since the reference points for oscillator U1C are closer together, the frequency increases.

Likewise, when the signal to U1B goes negative, the output of U1B increases and the output of U1A decreases. Since the reference points for oscillator U1C are farther apart, the frequency decreases. This is clearly seen in **Figure 6A**. The output from U1C is shown in **Figure 6B**.

The amplitude is always rail-to-rail/on and off, but the frequency changes based on the audio input signal. Diodes D1 and D2 make sure that the opposite polarity reference is used each time the U1C output switches from low to high or high to low.

The U1C output is fed to Q1-Q2 which forms a current mirror. R13 limits the current through Q2 to approximately 25-30 mA to drive the IR LED LD1. The current mirror assures that the LED output will be constant regardless of temperature or battery voltage (Vs).

The IR receiver shown in **Figure 7** is used to recover the original audio signal. The received FM pulses are fed to a PLL (phase locked loop) such as a CD4046. The PLL tries to match its frequency to the frequency of the incoming IR pulses. The phase error correction signal changes the local oscillator to match the incoming frequency.

The error correction signal is the same as the original audio signal which can be amplified and sent to a speaker.

Another use for the IR transmitter-receiver pair is for remote listening. To demonstrate this technique, get a small speaker and place a layer of aluminum foil over the frame of the speaker. Connect the speaker to an amplifier and play some music or a voice recording. Point the IR transmitter at the speaker. Do not connect any audio to the transmitter input. This gives a constant output frequency. Point the IR receiver at the speaker.

As the speaker diaphragm vibrates back and forth, so will the aluminum foil. As the IR waves reflect off of the aluminum foil, they will be frequency modulated by the vibrations due to the Doppler effect. The IR receiver will try to match the local PLL oscillator frequency to that of the incoming IR pulses, and you will hear the audio via the reflected IR beam.

Spies have used this technique to eavesdrop on conversations through windows which vibrate when sounds are made inside a room. They can remotely hear the conversation because of the vibrations of the window glass. (This would make a great science fair project.)

Now, with this information, you should be able to make light dimmers, PWM motor speed controllers, model train controllers, AM and FM modulated transmitters and receivers, heater controls, and a host of other applications. **NV**

Beyond the Arduino 7

It's All in the Timers

I have to own up — we've been writing terribly inefficient code in this series. That's right. Every time we've blinked an LED or read a temperature or logged an event, we've been wasteful. Thankfully, that is all about to change ...

It's All in the Timing

Time dictates how we live our lives. Most of us need to conform to a schedule specified by our working hours, meetings, train timetables, and kid's bath times. Even our recreation time is governed by the clock on the wall: the starting time of a sports game, arriving at the cinema in time for a film, not missing a dinner reservation.

It isn't that much of a jump to expect that our microcontroller projects are even *more* dependent on timing — albeit at a time scale that is magnitudes smaller than ours. Many of our projects so far have relied on this — baud rates, I^{2}C frequencies, even the simplest blinking of an LED all rely on accurate timing.

Given the importance of timing, we could reasonably expect that our microcontroller should have some fairly clever mechanisms to allow us to manage timing in our projects — and we would be right. This mechanism comes in the form of something called a timer.

A Timer! Of Course!

It sounds so logical — a timer to control the timing on your projects. So, why do we need a dedicated timer? We were doing just fine with our *_delay_ms()* functions that we were calling in earlier projects. Why would we want to complicate our lives further with another peripheral on our microcontroller?

To answer that question, let's ignore the other functionality that the timer provides and just focus on the ability to implement a delay for a specific period. The answer lies in the implementation of the *_delay_ms()* function. It is essentially a loop that "burns up" clock cycles in order to cause a delay for a specified period of time. So, if you ask for a one second delay, the MCU sits there spinning its wheels for one second counting out clock cycles — and all the while not able to perform any other function.

While it may sound like that is exactly what we want — the microcontroller to halt all other operations for that one second — it isn't really that efficient. Wouldn't it make much more sense to have the microcontroller carry on with other tasks, and then just tell us when the one second is up?

Often, the microcontroller may not have any other tasks to carry on with, in which case we could let it go to sleep (dramatically reducing its power usage) and then wake up when the one second is over. This is where timers come into play.

When I first made the transition from the Arduino environment to raw AVR programming, the concept of a timer made complete sense — until I started thinking through the mechanics: How does it keep time? How do I use it (Do I need to keep checking the time?)? What unit of time does it keep? What if I'm operating at a different clock speed?

So, of course, I headed to the datasheet, which (as was typical in my earlier days) threw me into total confusion. I poured myself a scotch and called it a night, with a plan to revisit the topic more cautiously the following day.

A Cautious Approach to Timers

The following day, I set the datasheet aside and began researching timers without getting into the specifics of Atmel's timers. I think that's a good place to start here too.

So, what exactly is a timer? It couldn't really get any simpler: A timer is a *counter* which increments (or in certain cases, decrements) in time with the CPU clock. The current count of the timer is stored in a register that we can access from our code. Sound useful? Not really!

By Andrew Retallack
www.crash-bang.com

Off the back of their basic counter functionality, timers generally offer a range of features that actually do make them useful.

Prescalers

We've dealt with prescalers in the past; they are like a gearbox for the microcontroller's clock, ratcheting down a fast clock into speeds that are more manageable. A 16 MHz clock ticks 16 million times a second, resulting in a time-slice that is far too small to be practical for most applications. A 1024 prescaler slows that down by a factor of (you guessed it) 1024, resulting in an achieved frequency of 15.6 kHz — far more useful (although still at a pretty high resolution). We'll be using the prescaler at its maximum value for this article.

Interrupts

Yes, interrupts come into play once again as an important tool. Timers are usually able to generate a number of different types of interrupts; the two most common being an overflow and compare interrupt. Let's look at the overflow first, as this is the simplest.

Remember how the current value of the timer is stored in a register and that the registers we've dealt with so far have been eight-bit registers that store a value from 0 to 255? So, what happens when the counter reaches 255? As with most registers (and variables in many languages), it "overflows" back to a value of zero and starts counting up again. You can configure the timer so that it generates an interrupt each time this overflow occurs.

The overflow interrupt is great, but only if you want to measure time in multiples of 1/61 of a second (that's how long it takes the timer to count up to 255 off a 16 MHz clock). What if you want to measure a different value? The answer is that you can set the timer to generate an interrupt at *any* value between 0 and 255 by setting a value in a compare register (on the ATmega328P, this is called an Output Compare register). If you set a value of 127 into this register, an interrupt will fire every time the timer count reaches 127.

Now, we've gained a bit of control and are actually getting somewhere!

Controlling a Pin

In addition to generating an interrupt, many timers can directly change the level of an output pin. This can be really useful if you want, for example, to have an LED flash away without any interruption or variation. Without putting any strain at all on your microcontroller, you can simply "set and forget" and the timer will make it happen.

Configure the registers and away you go. On the ATmega328P, there are only certain pins that this can happen on; those marked OC0A, OC1A, OC2A, OC0B, OC1B, and OC2B.

Time to Talk Frequency

When I first started exploring microcontrollers in general and timers in particular, I was taken back to my high school days as I began to delve into frequency and period. Perhaps a refresher will help you in the same way it did me.

Frequency is the number of times an event occurs in a specific time period. In the world of computing, we measure frequency in Hertz (Hz), which refers to the number of events per second. Our microcontroller runs at 16 MHz, meaning that its clock ticks 16 million times in a second. In our LED projects, we typically blink the LED once a second; in other words, at a frequency of 1 Hz.

Period (or the time taken for one clock tick to complete) is the reciprocal of frequency. If you divide "1" by a frequency, the result given is a duration in seconds; 16 MHz results in a period of 62.5 nanoseconds, or 0.0000000625 seconds. These are pretty short slivers of time!

A clock prescaler divides the frequency of a clock source (e.g., the main clock of your microcontroller) by a specific number. So, the 1024 prescaler will generate a "tick" for every 1024 cycles of the main clock. The result is that the frequency decreases to 15.625 kHz (16,000,000 / 1024). Remember that period is the inverse of frequency, so the decrease in frequency results in an increase in period. We now have a period of 64 microseconds (or 0.000064 seconds).

Timing an Event

Not only can the timer affect a change on a pin, it can also be used to detect a change on a pin. When a change on a pin occurs, the value of the timer is stored in a "capture" register, and an interrupt is generated to let you know that there's a value waiting to be processed. You could think of this functionality as a stopwatch for microcontrollers — useful for timing very small events; for example, measuring the frequency or time-based levels of an external component (ever wanted to build a basic digital oscilloscope?).

Generating PWM Signals

Timers are also able to generate PWM (pulse-width modulation) signals; in fact, this is how many microcontrollers implement PWM functionality. In the world of Arduino, we used the *analogWrite()* function to achieve this. Behind the scenes, however, it was all being run by the timers.

It is a little more complex in the "raw" world of AVR microcontrollers, but an understanding of the concepts will allow you to work with a much broader range of microcontrollers than the Arduino ecosystem allows.

Feature	Timer0	Timer1	Timer2
Precision	8-bit	16-bit	8-bit
Maximum Value	255	65,535	255
Output Compare	Yes – 2	Yes – 2	Yes – 2
Input Capture	No	Yes	No
PWM — Fast	Yes	Yes	Yes
PWM — Phase Correct	Yes	Yes	Yes
PWM — Phase & Frequency Correct	No	Yes	No
Asynchronous Operation	No	No	Yes

Table 1: Core timer features.

The Watchdog Timer

Finally (for now), we have the oddly-named watchdog timer. The main purpose of the watchdog is to keep an eye on your project, and initiate a reset if the project seems to be "hanging" in a non-responsive state. This is a good way to restart your project if it gets caught in an infinite loop — although, it shouldn't be relied on as a substitute for poor design and testing.

To implement the watchdog timer, you need to set a *timeout* value. The watchdog timer needs to be reset ("stroked" as it's commonly known) before the *timeout* value is reached. Otherwise, the microcontroller will be reset. There's more to watchdogs, but we won't touch on that right now.

Let's Get Specific

After that long list of features, let's take a look at how timers are implemented on our ATmega328P microcontroller. For this article, we're going to focus on the compare and overflow functionality of the ATmega328P timer; in future articles, we'll take a look at the capture and PWM features as I'd be doing them an injustice with a cursory discussion!

Three Timers to Play With

The ATmega places three different timers at our disposal: Timer0, Timer1, and Timer2 — meaning that we're able to have three sets of "timing" functionality running simultaneously. While in essence the three timers perform the same function, there are a few differences that may lead you to choose a specific timer for your project. Two of the timers (0 and 2) are eight-bit timers; they cover a shorter time period, overflowing after a value of 255. Timer 1 is a 16-bit timer, so it only overflows after 65,535. If we're running at 16 MHz with a 1024 prescaler, this translates into a possible time period of a little over four seconds.

Table 1 gives a quick comparison of the three timer's features.

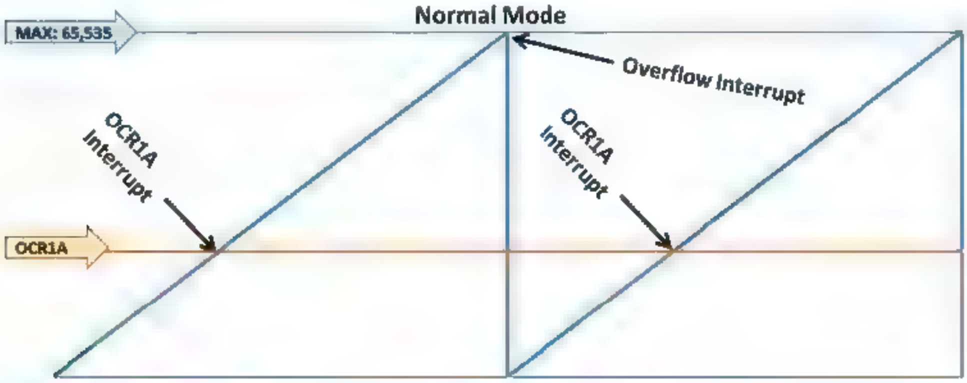


Figure 1: Timing diagram showing timer operating in Normal mode.

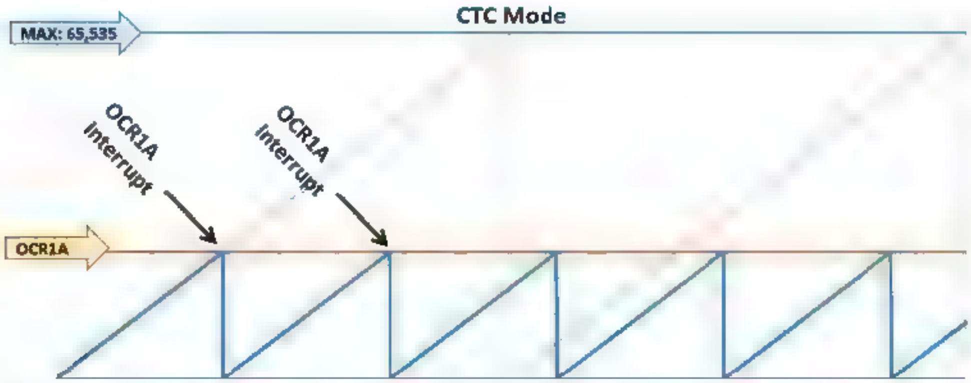


Figure 2: Timing diagram showing timer operating in CTC mode.

Deciding on the Mode

There are a number of modes that we can set the timers to run in, as **Table 1** hinted at. For this article, we'll focus on the two most common modes in order to establish a firm base; we'll then move on to others in future articles.

Normal Mode:

This is the simplest mode, starting at zero and counting up to the maximum values of 255 for eight-bit timers or 65,535 for 16-bit timers. Once the timer reaches the top, it simply overflows back to zero and starts counting up again, and again, and again.

Clear Timer on Compare (CTC Mode):

This mode is similar to the

Normal mode, except that it allows you to define the maximum value the timer counts up to, using the Output Compare register. This value, of course, needs to be less than or equal to the maximum that the timer can count up to. Once the Output Compare value is reached, the timer overflows to zero and counts up again.

For example, if you set Timer0's Output Compare register to 127, the timer would count up to 127 and then start from zero again. This is the mode that I normally use for any timing-related functionality in my applications, as it gives me control over the timer period.

Let me Inter...rupt You

It's one thing to set the timer's mode, but another to actually act on the timer values. This is where interrupts come into play. I mentioned earlier that we have two timer-related interrupts that we can choose to enable on the ATmega328P: the Overflow interrupt and the Output Compare interrupt.

The Overflow interrupt fires whenever the timer overflows the maximum timer value, i.e., 255 for Timer0 and Timer2, or 65,535 for Timer1. The Output Compare interrupt fires whenever the timer counts to the value in the Output Compare register. It's important to note that the Output Compare interrupt will trigger in both Normal and CTC modes. The difference is that in Normal mode, the timer will keep counting beyond the Output Compare value up to the maximum, whereas in CTC mode it will restart from zero.

If this all sounds a little confusing, then take a look at **Figures 1** and **2**. **Figure 1** shows the timer operating in Normal mode, while **Figure 2** shows the timer in CTC mode. You'll note that in Normal mode, the timer keeps incrementing all the way to 65,535 with both the Output Compare interrupt and the Overflow interrupt firing. In CTC mode, the timer has a shorter period, only counting up to the Output Compare value (OCR1A).

Additional Timer Features

There are a few features of timers that we won't cover in this article, but are worth touching on to give you more reasons to go exploring! The datasheet covers these items in more detail, but I typically find that I understand things better when I see working examples. Google is indispensable.

Clock Source

We've been driving the timers off the main system c ock, but they can alternatively be driven off external sources. This would allow you to, for example, synchronize with another project or module (e.g., a real time clock) that generates a square wave signal. Or, you could hook up a 555 timer at a frequency that you aren't able to generate with your ATmega, and use that to drive your timer. The clock source needs to be at least half the frequency of the system clock (ideally 2.5 times less), due to the way the ATmega samples the frequency.

Asynchronous Operation

Timer2 — which we haven't used yet — allows asynchronous operation. This means that it is able to run at a frequency that isn't linked to the main system clock — completely independently. This would normally be used for attaching an external 32 768 kHz watch crystal (the same type we used last month with our real time clock) which, of course, operates at a much lower (more useful) frequency.

The asynchronous operation is really useful when you start operating in low power modes. If your main clock is shut down to conserve power, your timer can keep operating off the external clock source. We'll cover low power modes in detail in a future article.

Time Registers

I know, time is marching on and we need to get started with this month's project (well, two projects actually). The final bit of detail we're going to look at is (as you probably expected) an overv ew of the registers involved. By now, you'll be pretty comfortable navigating the datasheet, but I still find it useful to highlight the registers that we'll be using in our projects. As each of the three timers have slightly different functionality, I've only summarized the registers for the 16-bit timer that we'll be using: Timer1. **Figure 3** contains this summary.

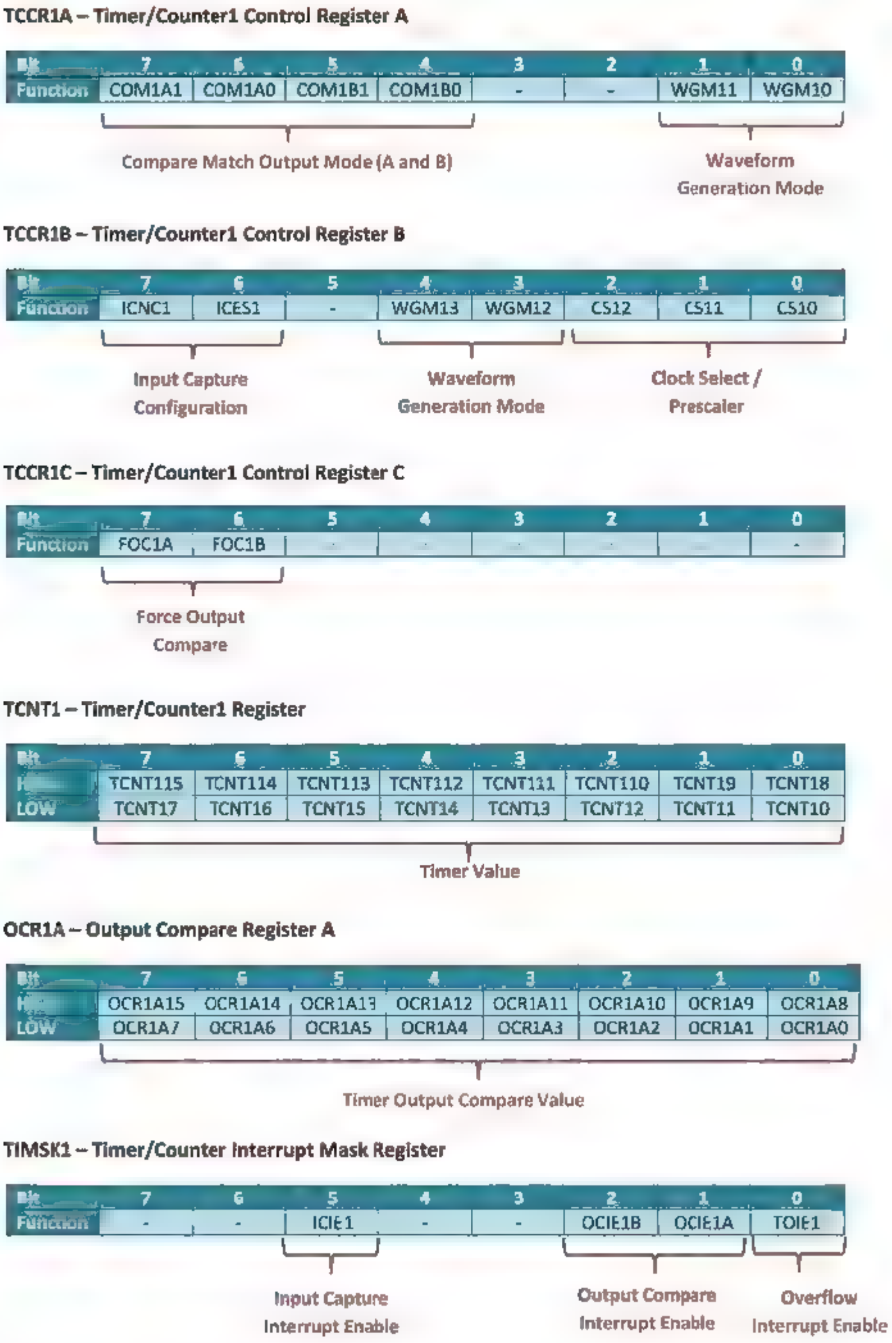
The easiest way to tackle the registers we need is to see them in context, so take a look at **Listing 1** in addition to the summary as we work through these. **Listing 1** contains the code we need to initialize the timer.

Control Registers

So far, we've seen that most peripherals have a number of control registers. Timers are no different. Timer1 has three control registers which configure — among others — the timer mode and the prescaler setting. Take a look at **Listing 1** with reference to **Figure 3**, and you'll see that we only need to use the Control Register B

Listing 1. Timer1 initialization routine.

```
void Timer_Init(void)
{

    TIMSK1 = 0;    //Disable all interrupts on
                   //timer

    TCCR1B  = (1<<CS12) | (1<<CS10);
                   //Set prescaler to 1024

    TCCR1B  = (1<<WGM12);    //Set to Clear-
                   //Timer-on-Compare-Match (CTC) mode

    OCR1A = 15625;
        // CPU Speed = 16MHz; Prescaler = 1024;
        //cycles per sec = 16000000/1024 = 15625

    TIMSK1 |= (1<<OCIE1A);
        //Enable interrupts on Compare
}
```



TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0
Function	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0
Function	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0
Function	FOC1A	FOC1B	-	-	-	-	-	-

TCNT1 – Timer/Counter1 Register

Bit	7	6	5	4	3	2	1	0
HIGH	TCNT115	TCNT114	TCNT113	TCNT112	TCNT111	TCNT110	TCNT19	TCNT18
LOW	TCNT17	TCNT16	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10

OCR1A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0
HIGH	OCR1A15	OCR1A14	OCR1A13	OCR1A12	OCR1A11	OCR1A10	OCR1A9	OCR1A8
LOW	OCR1A7	OCR1A6	OCR1A5	OCR1A4	OCR1A3	OCR1A2	OCR1A1	OCR1A0

TIMSK1 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
Function	-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1

Input Capture Interrupt Enable
Output Compare Interrupt Enable
Overflow Interrupt Enable

Figure 3: Summary of Timer1 registers.

(TCCR1B) to set the prescaler to 1024 and the mode to CTC. In setting the mode, we work with the *Waveform Generation Mode* bits (WGMxx). Don't be misled by the somewhat technical-sounding name; I know that I was initially confused.

Timer/Counter Register

The Timer/Counter register (TCNT1) contains the current value of the timer and also allows you to set the value of the timer. If for some reason you choose not to use interrupts for your timing and want to work off the underlying timer values, then this is where you'll find them.

You'll note in the datasheet and summary in **Figure 3** that the register has 16 bits. This is what we'd expect, given that Timer1 is a 16-bit timer. What we may struggle with, though, is working out how to set a 16-bit register on an eight-bit microcontroller. The Timer/Counter register is, in fact, made up of two eight-bit registers (TCNT1L and TCNT1H) containing a low byte and a high byte.

You may remember that we came across a similar conundrum back in the third article (*Nuts & Volts*, May 2015) when reading a 10-bit value off the ADC (analog-to-digital converter). Well, Atmel has kindly solved the Timer1 register issue in the same way as they solved the ADC issue: by providing a single register that *acts* as a 16-bit register. So, the answer to our puzzle is incredibly straightforward. We simply use the TCNT1 register for our read/write operations.

Output Compare Register

The Output Compare register contains (you may have guessed) the value that triggers the Output Compare interrupt. When the timer counts up to the value in the Output Compare register (OCR1A), then the interrupt will fire; assuming, of course, that it is configured to fire. This register needs to match the resolution of the timer.

So, for Timer1, it needs to be a 16-bit register. OCR1A is accessed in the same way as the TCNT1 register is: as a virtual 16-bit register and without the need to split the value being written into high and low bytes.

You may have noticed that we actually have two Output Compare registers — OCR1A and OCR1B. This sounds pretty interesting, so can we set two interrupts to be triggered off the same timer? Yes, we can, but only OCR1A can *clear* the timer if you have set it to run in CTC mode. This means that you would need to set OCR1B to a value lower than that of OCR1A.

Interrupt Mask Register

The final register we're going to look at now is the Timer/Counter Interrupt Mask register (TIMSK1). This is the register that we use to control which interrupts we want firing. By setting the appropriate bit to a 1, we can specify which of the Input Capture, Output Compare (A and B), and Overflow interrupts will be enabled. Very straightforward!

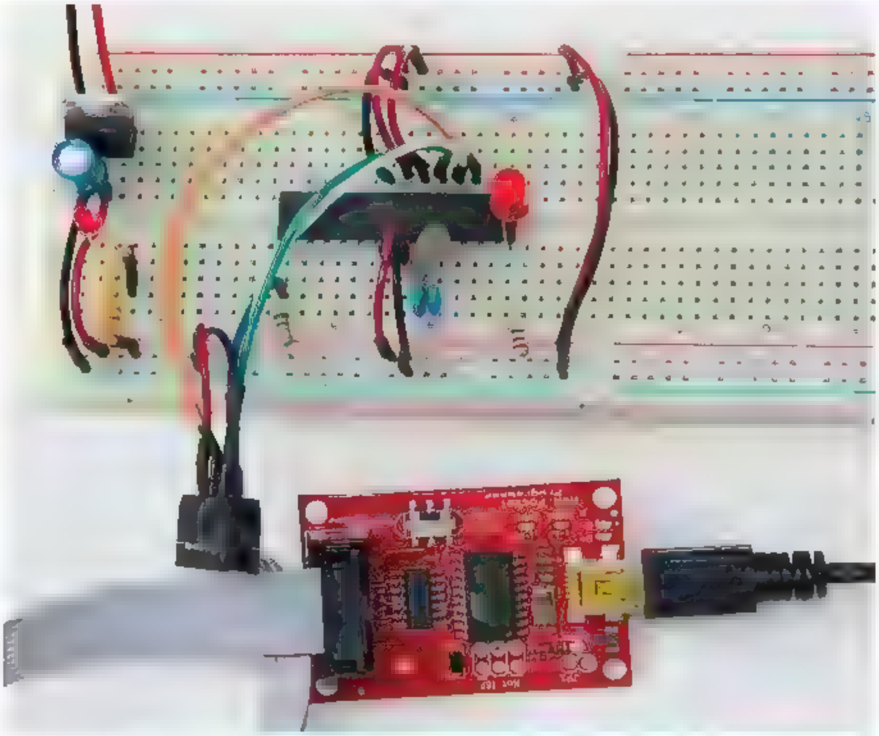


Figure 4: The "Blink" project laid out on a breadboard.

Bringing It All Together

Now that we've worked through the main registers we'll need to use, the code in **Listing 1** should make some sense. It is actually an extract from our first project this month: Blink

"Blink? Surely we've advanced beyond a project that simply blinks an LED?"

If you're thinking along those lines, you're 100% correct. We *have* advanced way beyond blinking an LED. However, through all our projects to-date, we've never blinked an LED *efficiently*. So, I'd like to go back to the roots of this series of articles and make that LED blink the way that all LEDs should blink!

You can download the project "Beyond Arduino 7 — 1 Timer Blink" at the article link. Open it up in Atmel Studio, and we'll take a quick look at how it works. **Figure 4** shows the project on a breadboard (the same setup as we had in April), while **Figure 5** shows an alternative using the Toadstool Mega328 (refer to **Resources**).

Initializing the Timer

We've spent some time looking at **Listing 1**, so we can run through the logic pretty quickly. As you'll see, this function is called before we enter the *while(1)* loop in our program.

First up, we disable all timer interrupts, just in case they were set previously. We don't want to be disturbed as we configure the timer. As the interrupt bits are all contained in the same register, we simply type:

```
TIMSK1 = 0;
```

The next statement sets the prescaler to 1024 — the maximum value. Take a look at Table 16-5 in the ATmega328 datasheet, and you'll see we need to set bits CS12 and CS10:

```
TCCR1B |= (1<<CS12)    (1<<CS10);
```

Next, we need to set the mode for the timer. From looking at Table 16.4 in the datasheet, it will be apparent that some modes require you to set both TCCR1A and TCCR1B. The CTC mode that we're using only needs WGM12 set:

```
TCCR1B  = (1<<WGM12);
```

With the mode set to Clear on Timer Compare, we, of course, need to set an Output Compare value. By dividing

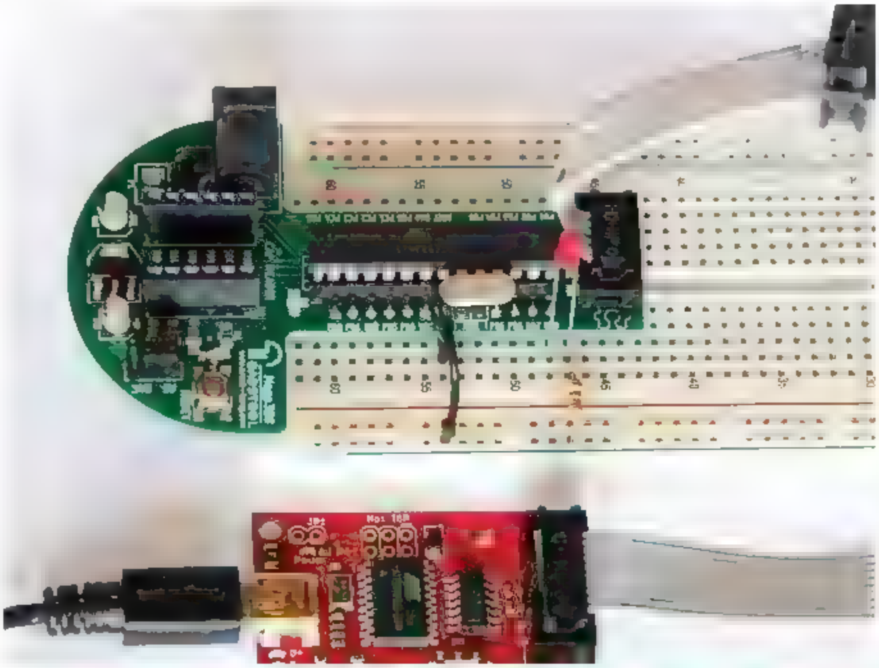


Figure 5: Using a Toadstool Mega328 for the "Blink" project.

the clock rate by the prescaler, we arrive at the number of "clock ticks" that will happen in one second: 16,000,000 / 1024 = 15,625. Let's use this value to make our LED flash every second:

```
OCR1A = 15625;
```

Finally, in order for us to actually do something with the timer, we need to enable an interrupt. As we set OCR1A with the compare value, we need to enable the equivalent interrupt:

```
TIMSK1 |= (1<<OCIE1A);
```

That's it! We've now configured Timer1 to trigger an interrupt every one second. The only other item remaining is to enable interrupts using the *sei()* command.

Handling the Interrupt

We spent the whole July article talking about interrupts, so this next interrupt service routine shouldn't look too foreign to you:

```
ISR(TIMER1_COMPA_vect)
{
    LED_PORT ^= (1<<LED_PIN);
        //Toggle LED
}
```

LED_PORT and *LED_PIN* are simply macros that have been defined as PORTB and PORTB0, respectively. I did this to make the code easier to alter. If I decide to change the pin that the LED is connected to, I only need to make the change once in the initial *#define* statement.

All this interrupt handler does is toggle the LED.

The Main *While(1)* Loop

Huh? Why is the *while(1)* loop empty? It's empty because of the beauty of using timers and their related interrupts. Everything is taken care of by the interrupt service routine which, in turn, is triggered by the timer running in the background. Elegant, don't you think?

Of course, it's unlikely that we'll build many projects that just flash an LED, but having the timer doing its thing in the background means that we don't need to worry about what else we've got to do in the main program. We can focus on the rest of the application, knowing that the timer interrupt will fire like clockwork and keep the LED flashing.

One extra comment: We can change the value of the Output Compare register (OCR1A) while the program is running. So you could, for example, make the LED flash faster and slower in response to some event.

A Second Timer Project

Now that you've managed to tear yourself away from that mesmerizing flashing LED, let's look at another little project. This project is fairly similar, so I've called it "Blink 2" (You'll have to pardon my South African creativity!).

This project is not so much a new project as an adaptation of the existing one. You can download it from the article link, but I'd prefer if you just went along with me to make the changes here. Firstly, why *another* blink project? Is this not getting a little tedious? Well, I'd like to demonstrate the ability for timers to interact directly with specific pins on the microcontroller, without the need for interrupts. Let's see that in action.

1. Move the LED.

In order to have real "hands-free" operation of the

```
#include <avr/io.h>

//Define the LED port and pin
#define LED_PORT        PORTB
#define LED_PIN         PORTB1
#define LED_PORT_DIR    DDRB

//Function Prototypes
void Timer_Init(void);

int main(void)
{

    //Initialise the IO Pins
    LED_PORT_DIR |= (1<<LED_PIN);
            //Set pin to Output
    LED_PORT |= (1<<LED_PIN);
            //Turn the LED on

    Timer_Init();

    while(1)
    {
    }
}
```

```
void Timer_Init(void)
{


    TIMSK1 = 0;
     //Disable all interrupts on timer

    TCCR1B |= (1<<CS12) | (1<<CS10);
     //Set prescaler to 1024

    TCCR1B |= (1<<WGM12);
     //Set to Clear-Timer-on-Compare-Match
     //(CTC) mode

    TCCR1A = (1<<COM1A0);
     //Toggle OC1A on Compare Match

    OCR1A = 15625;
     // CPU Speed = 16MHz; Prescaler = 1024;
     //cycles per sec = 16000000/1024 = 15625

}
```

Listing 2. Initialize Timer1 to toggle OC1A.

LED, we need to move the LED to pin **PB1**. Why PB1? Take a look at the Pin Configuration diagrams in section 1 of the datasheet, and you'll see that PB1 has the following in parentheses: **(OC1A/PCINT1)**.

We're interested in the OC1A portion of this, which means that this pin can be directly controlled by Timer1's Output Compare functionality. So, move the LED, and also change the definition of *LED_PIN* to:

```
#define LED_PIN     PORTB1
```

2. Re-initialize the Initialization Function.

We need to change a couple of things in the *Timer_Init()* function. Firstly, as we won't be using interrupts, you can delete the last line of the function that enables them.

```
TIMSK1 |= (1<<OCIE1A);
```

Secondly, we need to "connect" the timer to pin PB1. We do this using the Control Register A (TCCR1A). A look at Table 16-1 in the datasheet tells us we need the following line of code:

```
TCCR1A = (1<<COM1A0);
     //Toggle OC1A on Compare Match
```

3. No Interruptions, Please.

As we aren't using interrupts any longer, we can:

1. Delete the interrupt handler:
   `ISR(TIMER1_COMPA_vect)`
2. Delete the line that enables global interrupts:
   `sei();`
3. Delete the reference to the interrupt header file:
   `#include <avr/interrupt.h>`

Hands-Free Operation

We're done! **Listing 2** shows the final result. Our timer is now configured to flash the LED without needing any code to manipulate the I/O pin. While the code may not be as short as the initial Blink project we did back in April, it's infinitely more efficient and doesn't clutter up our *while(1)* loop.

What's Next?

I hope that you've found this discussion of timers to be interesting. Timers are very useful in embedded systems, and I incorporate them into my projects often. While we haven't delved into all areas, I think we've covered enough ground to get you up and running.

A Homework Challenge

If you want to explore timers more fully, why not try adapting the first "Blink" project from this article? Add a second LED into the project, and use Timer1's second Output Compare register (OCR1B) to make the LED flash twice as fast as the original one on pin PB0. You'll need to amend the initialization function, alter the existing interrupt handler for OCR1A, and add a new interrupt handler for OCR1B.

Resources

Author's website:
www.crash-bang.com

Toadstool Mega328:
www.crash-bang.com/toadstool
http://store.nutsvolts.com

Coming Up

I've had some great conversations with a number of you, and find it really interesting learning of the challenges that you face and the projects that you're working on. While I have a number of topics I still want to cover in this series, I'd love to hear if there are specific areas you'd be interesting in learning about. Please drop me a line to let me know your thoughts! **NV**

■ BY FRED EADY

Make Mine MURS

The IoT (Internet of Things) is hogging the spotlight here lately. That's a good thing, as I am a believer in using electronic devices for the betterment of mankind. I'm also for it because I love to design and build all things electronic. When you get right down to it, those IoT devices can't do much without their radio equipment. The "things" working and communicating out there in the IoT world are really cool. However, sometimes a "thing" may not be the thing you need to get it done. In some cases, all you really need is a good old fashioned radio to do the job.

MURS

What is MURS? Multi-Use Radio System is a communications method that is an unlicensed two-way radio service that operates under rules similar to those of today's CB radio. As long as you don't use a continuous carrier, you can talk and send data on it. MURS was established in 2000 and operates under Part 95 with a power limit of two watts.

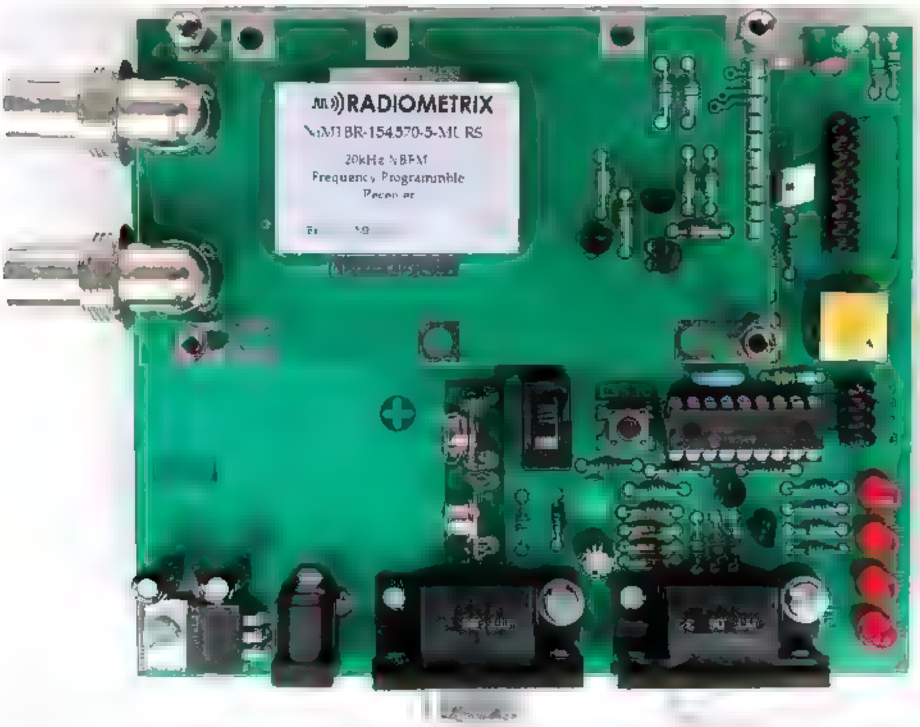
MURS is open to the public and can be used for personal and business communications tasks within the US. All commonwealth areas and islands under the control of the US are also legal. You can use MURS on a boat in international and domestic waters as long as the captain approves. Don't do MURS on an aircraft, and you can't take advantage of MURS on the public telephone network. You also can't set up a MURS repeater.

If you are a representative of a foreign government and have ideas about using MURS as a "spy network," don't get caught because that's not allowed. Building a "Hogan's Heroes" jail house MURS radio set is not permitted either. Only sanctioned and licensed MURS equipment can operate on the MURS frequencies. Don't feed a store and forward computing system with it either.

MURS is spread over five channels. The first three are narrow band FM channels with 2.5 kHz deviation. Channels 4 and 5 are wide band FM channels with 5 kHz deviation. The radios we will use in this project are "color dot" devices. That means their frequencies are designated by a channel name which is described as the color of a dot. Our radios fall into the Blue Dot frequency of 154.570 MHz (channel 4).

Our MURS system consists of a transmitter module and a receiver module mounted on a set of common carrier boards. The complete MURS receiver module is shown in **Photo 1**.

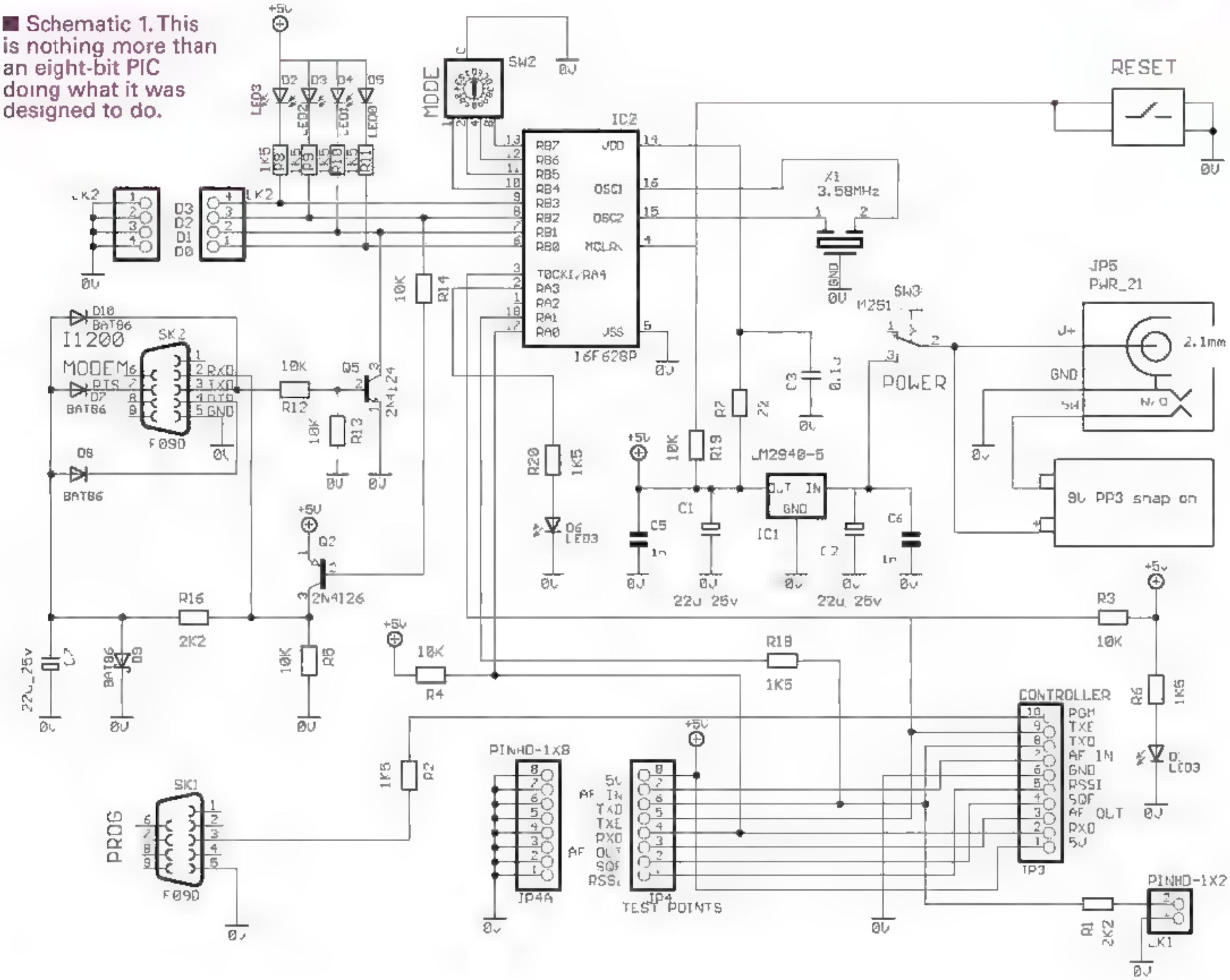
The communications protocol is implemented as half-duplex. There is no automatic collision detection. So, that's something we will have to handle either manually or via firmware in a host microcontroller.



■ Photo 1. Under the supervision of an eight-bit PIC, the MURS NBEK carrier board provides all of the power and data interface circuitry for the radio board.

NBEK Carrier Board

NBEK is short for Narrow Band Evaluation Kit. As you can see in **Schematic 1**, the NBEK carrier board is under the spell of a Microchip PIC16F628. The PIC is officially called the NBEK IC. The NBEK IC is



■ Schematic 1. This is nothing more than an eight-bit PIC doing what it was designed to do.

responsible for all transmit timing, valid data identification, and data stream buffering.

The NBEK IC is also programmed to act as a 1200 baud asynchronous modem. The modem acts as an interface to external data terminal equipment. Note the absence of the ever-present MAX RS-232 IC. Transmit data in the RS-232 format is converted to PIC logic levels by resistors R12 and R13, which are attached to NPN transistor Q5. Received RS-232 data is converted by PNP transistor Q2 and resistor R5. The target radio is attached at JP3.

You can easily pick out the voltage regulator circuitry which is based on an LM2940-5. Logic levels on the carrier board are at five volts. So, if a 3.3 volt radio is attached, five volt to 3.3 volt logic translation is necessary. Voltage for the NBEK carrier board and target radio is derived from an external power supply or a nine volt battery.

The NBEK carrier board circuitry is better understood when you can identify the operation of each of the PIC's I/O pins. Pin 1 is not connected. So, let's begin with pin 2. As you can see, pin 2 drives an LED. This pin is called the STATUS pin and is driven logically high when data is received or data is contained within the RX buffer.

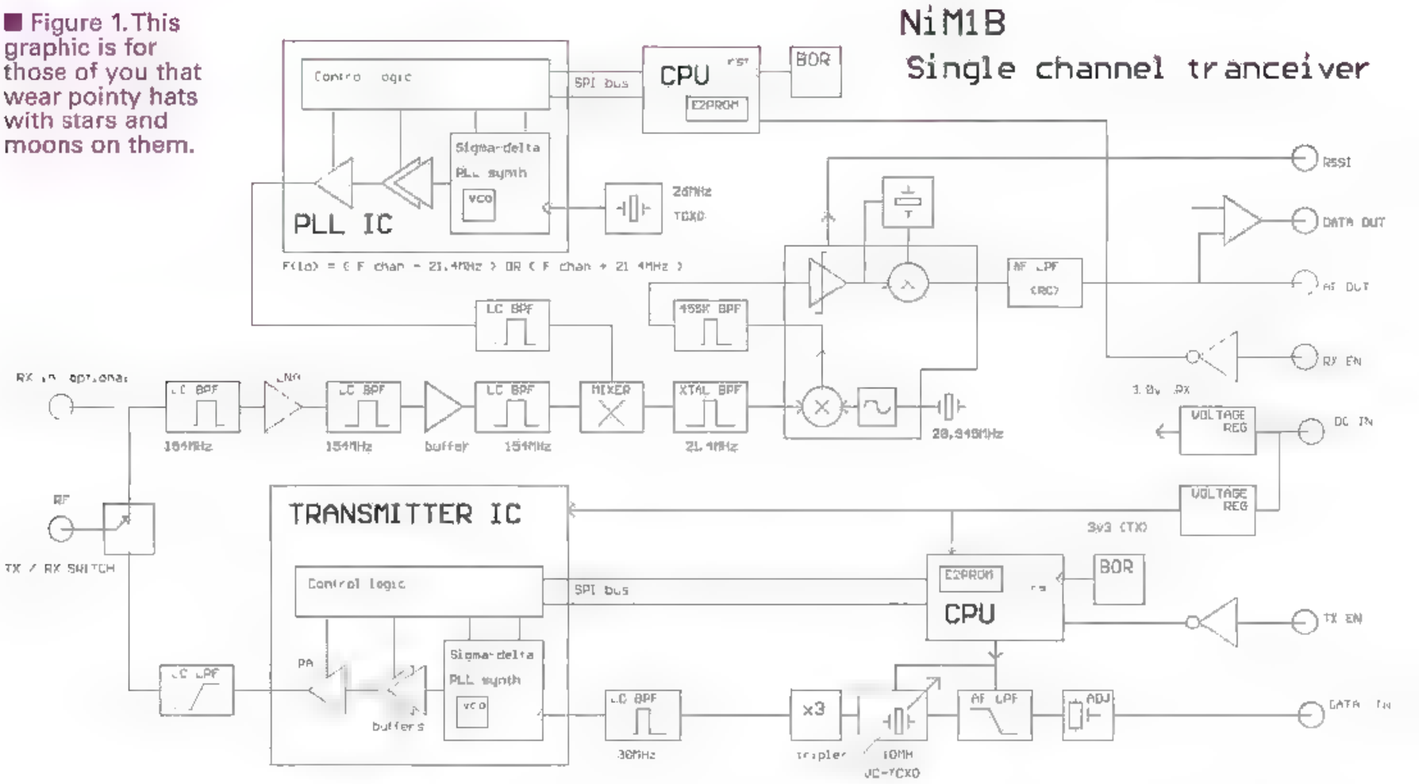
The NBEK IC automatically keys the attached transmitter when valid data is presented to the RXTXD pin (pin 7). The keying is performed by the active-low TXE pin (pin 3).

Pin 4 is the PIC's MCLR pin. For the purposes of the NBEK carrier board, it is identified as the RESET pin.

Pins 5 and 14 are Vss and Vdd, respectively. The PIC is powered by a five volt regulated power supply which is under the control of IC1.

Pins 6 through 9 are shared with the four-bit data word and the 1200 baud modem. D0 through D3 signals reside on pins 6 through 9, respectively. The modem transmit line is on pin 7. Pin 8 doubles as the modem receive pin.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/november2015_DesignCycle.**



■ Figure 1. This graphic is for those of you that wear pointy hats with stars and moons on them.

The function of the PIC's pins 6 through 9 is determined by the logic levels present on the Mode Select pins 10 through 13. The least significant bit of the Mode Select pins is pin 10. The logic levels of the Mode Select pins are determined by the position of rotary switch SW2. The Mode Select logic has a direct effect on the shared data I/O pins (6 through 9).

When in encoder/decoder mode, the D0-D3 pins are used for data entry/encoding (transmitter) and data reception/decoding (receiver). LK2 is used to determine the data encoding. A jumper on one of the LK2 data pins sends a logical zero to the receiver for the jumpered bit position. Obviously, the receiver LK2 pins should all be left open as they will reflect the state of the transmitter's LK2 jumpers.

The NBEK carrier boards and associated radios can be used as a remote control transmitter/receiver pair when the transmitter Mode Select switch is at position 8 and the receiver Mode Select switch is at position A. Placing both the transmitter and receiver Mode Select switches in the E position invokes the 1200 baud modem mode. Both the transmitter and receiver LK2 jumper blocks should be cleared in 1200 baud modem mode. This allows the NBEK carrier board's LED banks to indicate transmit and receive activity.

Test and measurement modes of operation supported by the NBEK IC include square wave modulated transmission, unmodulated transmission, and pseudorandom NRZ stream modulation.

Rotating the Mode Select hexadecimal rotary switch a bit further powers the NBEK carrier board up and down every 100 mS. A few more clicks on the transmitter's Mode Select switch automatically transmits the status of the LK2 data jumpers. Likewise, those clicks on the receiver enable the automatic reception of the transmitter's LK2 jumper status. If a NiM1B transceiver is being hosted by the NBEK carrier board, a master/slave echo mode can be dialed up.

The PICs on the NBEK carrier boards are clocked at 3.58 MHz by three-pin resonators on pins 15 and 16. The remaining pair of PIC I/O pins is used to provide a five volt logic level to the target radio.

The NiM1B

The NiM1B is a narrow band FM VHF radio. With the correct antenna configuration, the NiM1B can obtain ranges up to one kilometer. Our NBEK carrier boards are loaded with dedicated receiver (NiM1BR) and transmitter (NiM1BT) radio modules. A transceiver version of the

NiM1B is also available. The NiM1B family of VHF radio modules can support a data rate of up to 5 kbps.

The NiM1B datasheet claims that the radio modules can be reprogrammed on any frequency between 120 MHz and 175 MHz. The NBEK carrier boards have a PROG connector. However, the only mention of reprogramming in the datasheet points the user to the factory for such operations.

The NiM1B transmitter subsystem is based on a sigma delta synthesizer-based RF device, which can be configured via an SPI portal by an internal microcontroller. Upon assertion of the active-low transmit enable signal, the NiM1B can transition to full output power within 5 ms.

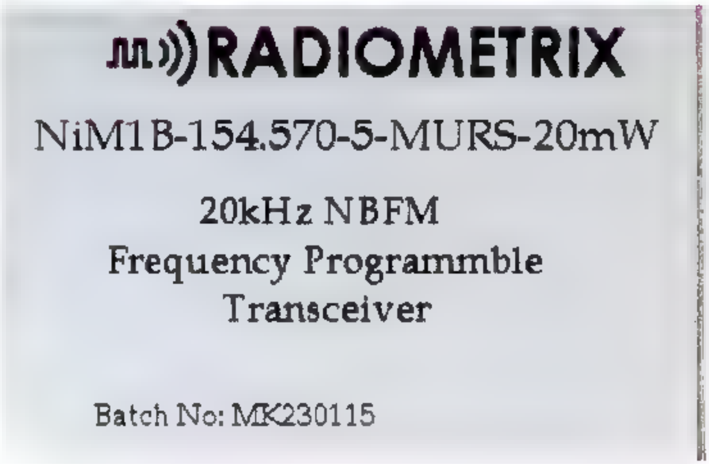
The NiM1B receiver subsystem is based on a sigma delta synthesizer-based local oscillator that can also be configured via an SPI portal by the internal microcontroller. The NiM1B receiver is a conventional superhet design with a MOSFET mixer and crystal/ceramic filter. I've included a block diagram of the NiM1B transceiver in **Figure 1**. **Figure 1** also applies to the dedicated receiver and dedicated transmitter variants of the NiM1B.

In that we have two NBEK carrier boards that are not loaded with a transceiver module, we can only collect data at the receiving station. If our NBEK carrier boards were loaded with the transceiver version of the NiM1B (like the one you see in **Photos 2** and **3**), we could communicate station-to-station in half-duplex mode. Recall that half-duplex mode is mandatory in that there are no automatic collision detection methods in place for the NiM1B.

Interfacing the NBEK Carrier Board Modem

The NBEK carrier boards are hard-wired to accept true RS-232 signals at the 1200 baud modem input. If you take another look at **Schematic 1**, you will see that the electronics behind the modem's TXD and RXD lines are wired as a null modem. That's a fancy way of saying the NBEK modem's RXD pin is actually sending to the other device while its TXD pin is really receiving from the attached device. The null modem wiring scheme allows us to tie the NBEK 1200 baud modem directly to an external device's serial port that is configured as DTE (Data Termina Equipment).

A good example of a DTE serial port is the nine-pin male serial ports that used to be found on PCs prior to the adoption of USB.


RADIOMETRIX

NiM1B-154.570-5-MURS-20mW

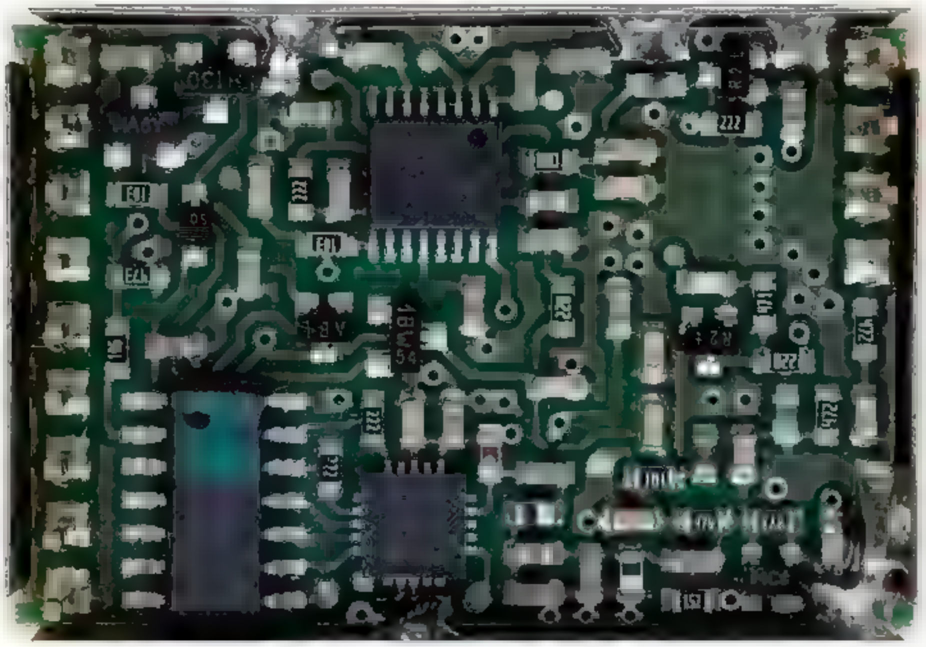
20kHz NBFM
Frequency Programmble
Transceiver

Batch No: MK230115

■ Photo 2. This aerial view captures the top panel of the NiM1B narrow band FM VHF transceiver. The NiM1B measures in at 33 mm x 23 mm x 11 mm.

The NBEK's null modem hookup lends itself to attachment to embedded RS-232 serial ports, as well. The embedded platform you see in **Photo 4** happens to be equipped with a true RS-232 serial port in the form of an SI3232C RS-232 transceiver IC.

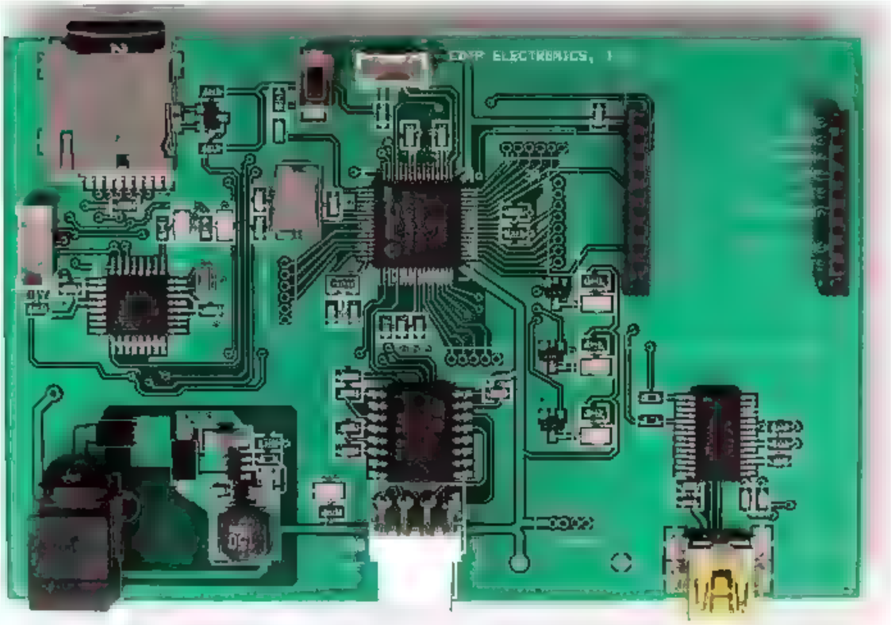
In reality, the NBEK's modem interface is actually operating at TTL logic levels. From the modem's perspective, Q5 can be driven with a five volt signal. In fact, it can be driven with a 3.3 volt logic level, as well. With its emitter tied to +5 VDC, transistor Q2's collector will toggle between 0 and +5 VDC. If we were to attach a 3.3 volt microcontroller UART directly to the NBEK's modem interface, we would only have to convert the voltage generated by Q2 That can easily be done with a 74LVC1T45.



■ Photo 3. This is a more interesting view of the NiM1B.

Many of the peripheral devices that use a serial port for communications have permanently set baud rates. If the device we wish to attach doesn't run at 1200 bps, we'll have to perform a baud rate conversion. The easiest way to do that is to use a microcontroller that supports multiple UARTs. Here's how it can be done with the PIC32MX575F512L that is hosting the board pictured in **Photo 4**:

```
//Initialize UART1 for 1200 bps
U1BRG = 4166; // Set Baud rate 1200
U1STA = 0;
U1MODE = 0x00008000; // Enable UART for
                     // 8-none-1
U1STASET = 0x00001400; // Enable Transmit and
                       // Receive
//priority 2 sub priority 3 - 00011111 bits
//<4:2> sub bits <0:1>
IPC6SET = 0x0000000B;
IEC0SET = 0x08000000;
IFS0CLR = 0x08000000;
// flush receive buffer
USART1_RxTail = 0x00;
USART1_RxHead = 0x00;

//Initialize UART2
U2BRG = 520; // Set Baud rate 9600
U2STA = 0;
U2MODE = 0x00008000; // Enable UART for
                     // 8-none-1
U2STASET = 0x00001400; // Enable Transmit and
                       // Receive
//priority 2 sub priority 3 - 00011111 bits
//<4:2> sub bits <0:1>
IPC8SET = 0x0000000B;
IEC1SET = 0x00000200;
IFS1CLR = 0x00000200;
// flush receive buffer
USART2_RxTail = 0x00;
USART2_RxHead = 0x00;
```



■ Photo 4. If this looks familiar, it is. It's the general-purpose 32-bit PIC32MX575F512L board we assembled in a previous Design Cycle. It is equipped with a true RS-232 port, a USB portal, and an intelligent microSD card subsystem.

The PIC32MX575F512H's UART1 interfaces to the NBEK modem. The second PIC32MX575F512H UART attaches to the peripheral device which, in this case, is running at 9600 bps. Each of the UARTs is supported by an interrupt-driven buffer. The idea is to take any incoming data from the peripheral device into UART2 and pass it as quickly as possible to UART1:

```
do{
if(CharInQueue2())
 {
   U1TXREG = recvchar2();
   while(U1STAbits.TRMT == 0);
 }
}while(1);
```

Each time a character is taken in and stored by UART2 — which is running at 9600 bps — it is retrieved from the UART2 receive buffer and placed in the transmit buffer of UART1, which is running at 1200 bps.

This same baud rate conversion algorithm can be performed at the receiving end. All we have to do is turn around the data paths:

```
do{
if(CharInQueue1())
 {
   U2TXREG = recvchar1();
   while(U2STAbits.TRMT == 0);
 }
}while(1);
```

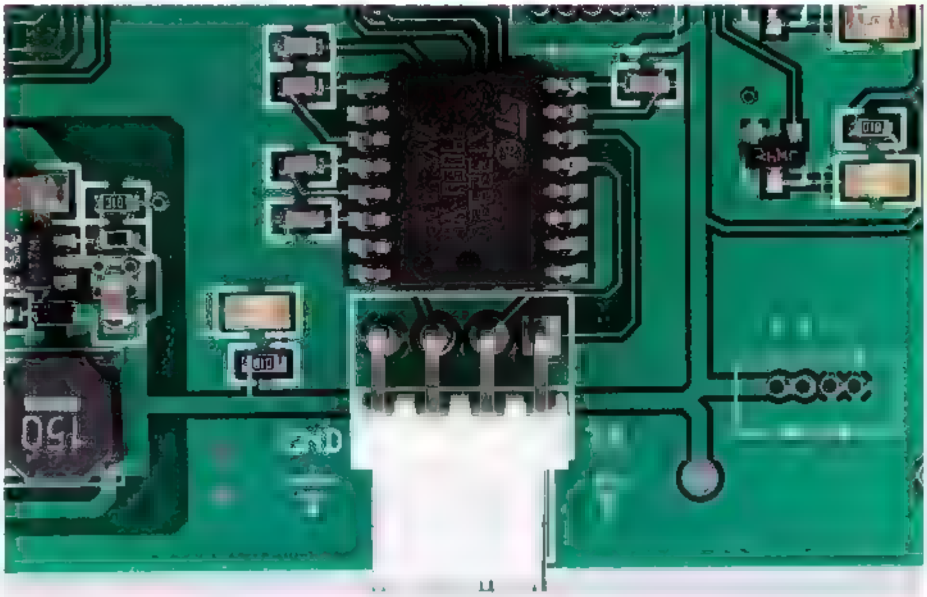
Assuming the use of the same PIC32MX575F512H hardware on the receiving end, we would convert 1200 bps data coming in from the NBEK modem to 9600 bps data for the attached peripheral device.

Interfacing the NBEK Data I/O Pins

The NBEK IC is preprogrammed to transmit and receive the status of the four LK2 pins (D0-D3). With no processing of the data, that gives us four on/off control pins. With the help of a PIC, the four I/O signals can be expanded to 16 I/O signals. For this example, let's use a PIC16F628 with the D0-D3 I/O lines connected just as they are in **Schematic 1**. Our C compiler of choice is CCS C:

```
#include <mux628.h>
unsigned int8 data;

void main()
{
  while(TRUE)
  {
   data = input_b();
   //read PORTB
   data &= 0x0F;
   //save least significant
```

```
    //nibble
    switch(data)
    {
      case 0x00:
       // all relays OFF
      break;

      case 0x01:
       //relay 1 ON
      break;

      // case 0x02 - case 0x0E
      // statements go here

      case 0x0F:
       //relay 15 ON
      break;
    }
  }
}
```



■ Photo 5. The ST3232C only needs a few charge capacitors to convert microcontroller UART logic levels to RS-232 marks and spaces.

This CCS C receiver application drives 15 relays. The lower four bits of PORTB represent the transmitter LK2 jumper status. PORTB is read and the upper four bits are forced to zero. The PORTB nibble is saved in the variable *data*. One of 15 relays is activated depending on the value of *data*. Here, we're controlling relays and the PIC16F628 doesn't have enough I/O pins to directly control all 15. With that, the functions performed within the 16 *case* statements can be anything your application requires.

For the transmitter application example, we'll have to move our PIC16F628's data I/O pins to the lower nibble of PORTA. The PIC16F628's UART pins occupy some of the PORTB pins:

```
#include <mux628.h>

#use delay(crystal=3580000)
#use rs232(baud=9600,parity=N,xmit=PIN_B2,
rcv=PIN_B1,bits=8,stream=PICUART)

unsigned int8 data;

void main()
{
  while(TRUE)
  {
    if(kbhit())
    {
            data = fgetc(PICUART);
          //get serial port data
            switch(data)
            {
                  case 0x30:
                //ASCII character '0'
                      output_a(0x00);
                  break;

                  case 0x31:
                  //ASCII character '1'
                      output_a(0x01);
                  break;

                  // case 0x32 - case 0x45
                  // statements go here
                  // 2,3,4,5,6,7,8,
                  // 9,A,B,C,D,E

                  case 0x46:
                  //ASCII character 'F'
                      output_a(0x0F);
                  break;
            }
        }
    }
}
```

The CCS C transmitter mux code takes its input from the PIC16F628 serial port. If an ASCII character equivalent of a hexadecimal number is received, the associated case statement will set the transmitter's LK2 pins with the specified binary value. Using this code, the NBEK transmitter's LK2 jumper logic can be controlled from a terminal emulator.

Combined with the code we wrote for the receiving carrier board's external PIC, ASCII characters generated at the transmitting NBEK carrier board by its external PIC are translated into actions at the receiving NBEK carrier board.

The Link is Up

The NBEK carrier boards are loaded with MURS radios and the NBEK IC enables reliable long distance data communications. As we have seen, the NBEK carrier boards allow for easy interfacing to external microcontrollers and other peripherals by way of the 1200 bps modem and LK2 jumper block.

We worked with the 20 mW versions of the NiM1B radios. Just think how far you could go with a two watt version. **NV**

ELECTRONET

For complete product details, visit our webstore!!

The Nuts & Volts WEBSTORE

GREAT FOR DIYers!

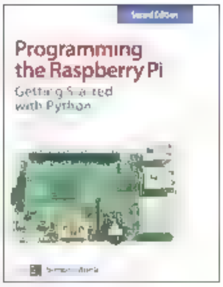
Beginner's Guide to 3D Printing by Chuck Hellebuyck

This book was written to answer the questions most beginners need answered. It covers many of the popular 3D printer choices and then uses the under $500 Da Vinci 1.0 from XYZprinting to show how easy it is to get started. Chuck takes you through using Tinkercad software for creating your own custom designs, then goes further and shows you how to take a simple design and send it off to a profess onal 3D printer This book was designed for anyone just getting started. **$24.95**

Electronics Workshop Companion for Hobbyists by Stan Gibilisco

In this practical guide, electronics expert Stan Gibilisco shows you step-by-step how to set up a home workshop so you can invent, design, build, test, and repair electronic circuits and gadgets. *Electronics Workshop Companion for Hobbyists* provides tips for constructing your workbench and stocking it with the tools, components, and test equipment you'll need. Clear illustrations and interesting do-it-yourself experiments are included throughout this hands-on resource.
$25.00

Programming the Raspberry Pi, Second Edition: Getting Started with Python by Jack Simon



This practical book has been revised to fully cover the new Raspberry Pi 2, including upgrades to the Raspbian operating system. Discover how to configure hardware and software, write Python scripts, create user-friendly GUIs, and control external electronics. DIY projects include a hangman game, RGB LED controller, digital clock, and RasPiRobot complete with an ultrasonic range finder.

$15.00

Make Your Own PCBs with EAGLE by Eric Kleinert

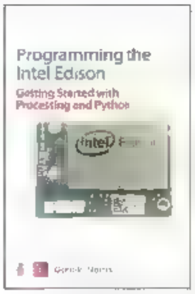
Make Your Own PCBs with EAGLE
From Schematic Designs to Finished Boards

Featuring detailed illustrations and step-by-step instructions, *Make Your Own PCBs with EAGLE* leads you through the process of designing a schematic and transforming it into a PCB layout. You'll then move on to fabrication via the generation of standard Gerber files for submission to a PCB manufacturing service. This practical guide offers an accessible, logical way to learn EAGLE and start producing PCBs as quickly as possible

$30.00

Programming the Intel Edison: Getting Started with Processing and Python by Donald Norris

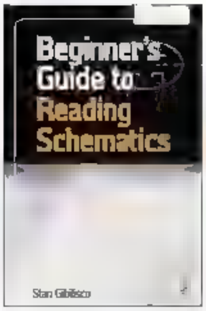
Learn To Easily Create Robotic, IoT, and Wearable Electronic Gadgets!


Programming the Intel Edison
Getting Started with Processing and Python
Donald Norris

Discover how to set up components, connect your PC or Mac, build Python applications, and use USB, Wi-Fi, and Bluetooth connections. Start-to-finish example projects include a motor controller, home temperature system, robotic car, and wearable hospital alert sensor

$20.00

Beginner's Guide to Reading Schematics, 3E by Stan Gibilisco

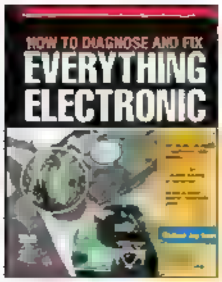


Navigate the roadmaps of simple electronic circuits and complex systems with help from an experienced engineer. With all-new art and demo circuits you can build, this hands-on, illustrated guide explains how to understand and create high-precision electronics diagrams. Find out how to identify parts and connections, decipher element ratings, and apply diagram-based information in your own projects.

$25.00

How to Diagnose and Fix Everything Electronic by Michael Jay Geier

Master the Art of Electronics Repair



In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.
$24.95

Programming PICs in Basic by Chuck Hellebuyck

If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

Programming Arduino Next Steps: Going Further with Sketches by Simon Monk

In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Also shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download. **$20.00**

EDUCATIONAL

Beginners Guide Book Combo.

Only $85.95
Plus
FREE Priority Mail Shipping
US Only

DO YOU LOVE RADIOS?

Build Your Own
TRANSISTOR RADIOS
Ham and Shortwave Radio for the Electronics Hobbyist
ARDUINO PROJECTS for AMATEUR RADIO

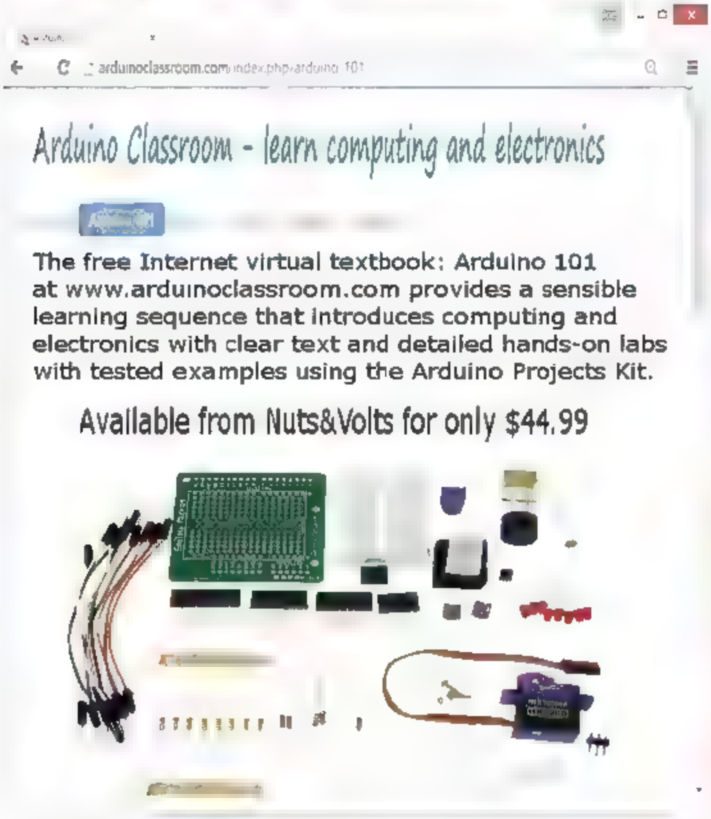
Sale Price $39.95

Sale Price $19.95

Sale Price $23.95

Get 20% off these three books

To order call 800 783-4624 or visit: http://store.nutsvolts.com



Arduino Classroom - learn computing and electronics

The free Internet virtual textbook: Arduino 101 at www.arduinoclassroom.com provides a sensible learning sequence that introduces computing and electronics with clear text and detailed hands-on labs with tested examples using the Arduino Projects Kit.

Available from Nuts&Volts for only $44.99

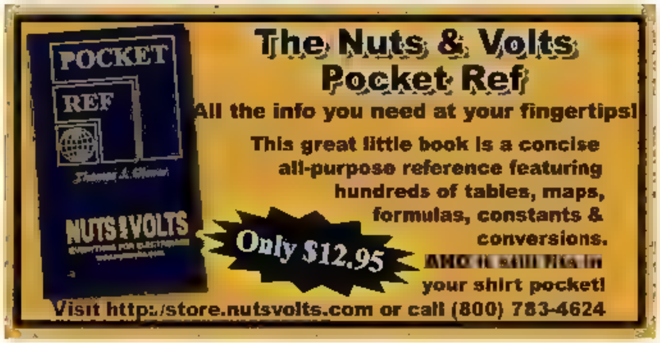
CD-ROM SPECIAL

Nuts & Volts 11 CD-ROMs & Hat Special!

That's 132 issues.
Complete with supporting code and media files.

Free Shipping!

Only $229.95
or $24.95 each.



The Nuts & Volts Pocket Ref

All the info you need at your fingertips!

This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants & conversions.

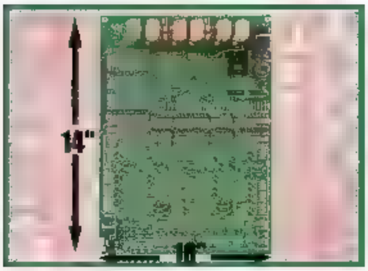
Only $12.95

AND it still fits in your shirt pocket!

Visit http:/store.nutsvolts.com or call (800) 783-4624

PROJECTS

Neon Transistor Clock Kit



Add HIGH VOLTAGE to your clock!

This is a Nixie Tube display version of the Transistor Clock. It uses only discrete components — no integrated circuits. Approximatly 1,250 of them! This kit is a "must have" for the analog electronic hobbyist. For more info, see the April 2012 issue or download the 87 page manual from our webstore.

Sale Price **$245.95**

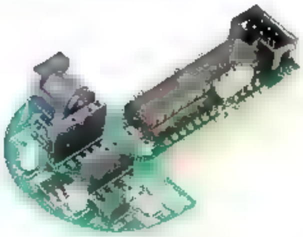
Seismograph Kit

Now you can record your own shaking, rattling, and rolling.

The Poor Man's Seismograph is a great project/device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

$79.95

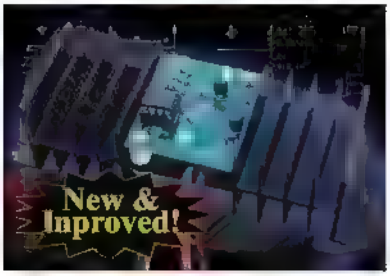
Toadstool



The Toadstool Mega328 was developed to assist with difficult breadboard projects. It is a prototyping board containing the Atmel ATmega328P microcontroller and allows you to build flexible projects right on your breadboard. It is ideal for hobbyists and enthusiasts looking for a more enjoyable microcontroller experience!

$29.95

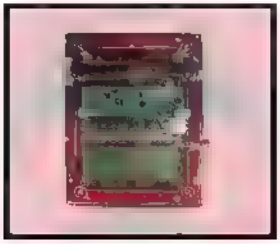
Solar Charge Controller Kit 2.0



If you charge batteries using solar panels, then you can't afford not to have them protected from over-charging. This 12 volt/12 amp charge controller is great protection for the money. It is simple to build, ideal for the novice, and no special tools are needed other than a soldering iron and a 9/64" drill!

$27.95

Geiger Counter Kit



This kit is a great project for high school and university students The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr. The LND 712 tube in our kit is capable of measuring alpha, beta, and gamma particles.

Partial kits also available.

$159.95

3D LED Cube Kit

This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes. Colors available: Green, Red, Yellow & Blue

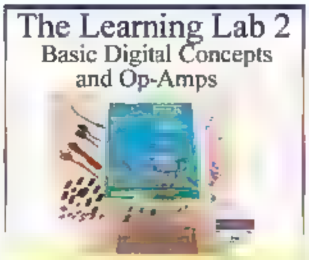
$57.95

FOR BEGINNER GEEKS!

The Learning Lab 1
Fundamental Concepts

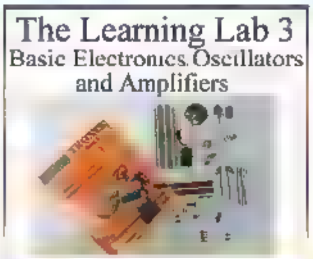
$59.95

The Learning Lab 2
Basic Digital Concepts and Op-Amps



$49.95

The Learning Lab 3
Basic Electronics Oscillators and Amplifiers



$39.95

These labs from LF Components show simple and interesting experiments and lessons, all done on a solderless circuit board.

As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and lab details, please visit our webstore.

■ BY WARD SILVER N0AX
hwardsil@gmail.com

Protecting Inputs and Outputs

Don't let a temporary glitch do permanent damage!

Part of the fun of ham radio is not only building your own equipment, but using it in all sorts of interesting places. For example, being able to provide portable and mobile communications after a disaster strikes your community often depends on your being able to put together an effective station in the field — away from your usual station or "shack."

Being out and about also means you are exposed to the various insults and accidents that lead to the infamous "Murphy bites." For a mobile installation, the electrical bus of a vehicle is home to a collection of transients and noise. Power connections can be mismade with surprising ease in any situation.

This column surveys some of the *mitigations* hams use against gremlins waiting just on the other side of the connection. Let's start at the innocent power jack, shall we?

Power Protection

We've all done it — accidentally connected power with plus where minus should be and vice versa "just for a second" — letting the magic smoke out of our precious components. For example, it is very, very easy to accidentally connect the wrong terminals of a 9V battery to its snap-on clip. Oops!

Addressing that very problem, Terry Fletcher WAØITP, published a collection of circuits in *The QRP Quarterly* to protect equipment against reversed power polarity.[1] I thought these were so useful, I added them to the *ARRL Handbook*, too![2] Most of these circuits can be retrofitted into existing equipment — as long as you account for the strengths and weaknesses of each circuit.

Diode Protection

Figure 1A shows the simplest protective circuit of all: a reverse-polarity blocking rectifier in series with the positive lead between the power source (in) and electronic device (out). All diodes connected to external wiring should have a small parallel capacitor to prevent RF from being rectified, which can generate EMI to other devices and the equipment being protected. (Be sure to check out this month's Open Communication column which talks about EMI/RFI extensively.)

The typical 1N4000-series rectifier has a nominal forward voltage drop (V_F) of 0.7V, but at *A* of current, it can grow to more than 0.9V with nearly 1W of power dissipated as heat. Schottky barrier rectifiers — shown in parallel to the conventional junction rectifier in **Figure 1A** — have a lower forward voltage drop: 0.75V at 1A for the 1N5418.

Be sure your equipment will work properly with its input voltage reduced a little bit by the diode. Don't exceed the diode's average forward current ($I_{F(AV)}$) rating or it will overheat and fail. The protective rectifier should have a reverse voltage rating (I_{RRM}) of at least twice the supply voltage, so a 1N4002 (50V, 1A) or 1N5418 (30V, 1A) are suitable in 12V systems.

The diode bridge circuit in **Figure 1B** not only protects the circuit, but allows it to operate normally with power voltage of either polarity applied. (You can also use AC power if the input circuit includes filtering and regulation.)

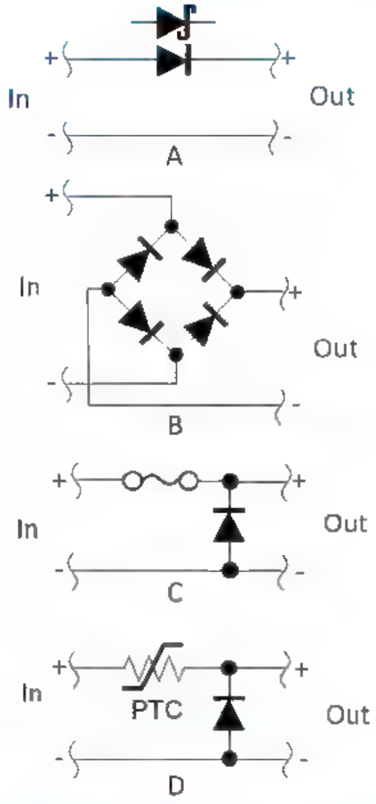


FIGURE 1. Diodes connected in series and a shunt can either block reverse polarity voltage or cause a protective fuse to open if voltage is reversed (see text). All diodes should be bypassed with a small value capacitor (such as a 100 pF - .01 µF, 50V ceramic) to prevent RF rectification and EMI generation.

The tradeoff is there are two voltage drops in series with the power source, so your operating voltage takes a hit of 1.5V to 2.0V, depending on what type of diodes you use. Diode bridges can be purchased all in one package, so this is a convenient option.

Figures 1C and **1D** rely on a fuse

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/november2015_HamWorkbench.**

ATC or ATO Fuse?

The fuses supplied with most mobile radio equipment are ATC style. Most automotive fuses are ATO. The ATC fuse element is completely sealed in plastic and the ATO is not. Since the power cable fuse holders are not waterproof, only an ATC fuse should be used if the fuseholder is exposed to the weather or located anywhere in an engine compartment. If an ATO type is used and water gets into the fuse, the fuse element corrodes and eventually fails.

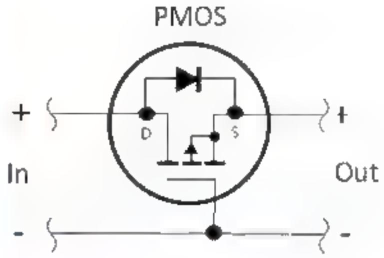


FIGURE 2. A PMOS enhancement mode MOSFET conducts with a very low on-resistance with normal voltage polarity, but blocks reverse polarity voltage.

to protect the circuit. A shunt-connected diode across the power source conducts high current if voltage polarity is reversed.

This will blow a metal fuse very quickly, and the diode will limit any applied voltage to the negative value of the diode's forward voltage. The fuse has no significant voltage drop under normal conditions.

The fuse needs to be large enough to handle normal device current, but not so large that the surge current through the reverse connected diode causes the diode to fail first. If you use this method, make sure the fuse is accessible and replaceable.

Be sure to use a fuse type appropriate for the environment as described in the **sidebar.**

The PTC (positive temperature coefficient) fuse in **Figure 1D** is a resettable fuse that acts like a solid-state circuit breaker. When its rated current is exceeded, the fuse material heats up and changes to a high resistance state. When the material cools, it changes back to the low resistance state.

PTC fuses are specified by a trip current at which the material changes resistance and their holding current that keeps them in the high resistance state.

A typical PTC device for low current 12V protection is the Littelfuse 60R050XPR, which has a trip current of 1A and a holding current of 500 mA. PTC fuses do heat up a bit when in the high resistance state, and they insert a small resistance in series with the power source (around 1W) that can cause a voltage drop.

Sometimes the shunt diodes in **Figures 1C** and **1D** fail due to the high surge current during a polarity fault. If a diode with an insufficient forward surge current (I_{FSM}) is used, it might blow the fuse but also fail as an open circuit or high resistance, leaving you unprotected in the future.

If shunt diode protection is used and the fuse opens, check the diode to be sure it has not failed as well. If the diode fails shorted or in a low resistance state, the fuse will continue to blow until the diode is replaced.

Active Device Protection

Many voltage regulators that use PNP or PMOS transistors (P-channel MOSFETs) such as the LM2940 offer reverse polarity protection. A PMOS transistor will also work as shown in **Figure 2.** (The diode symbol represents the intrinsic body diode that is created due to the way the transistor is made.)

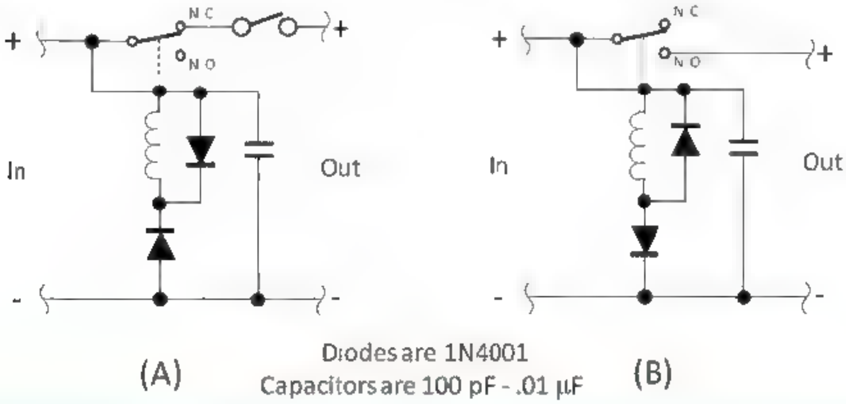
The PMOS enhancement-mode transistor conducts unless the drain-to-gate bias becomes negative as when the applied voltage is reversed.

PMOS devices have low on-resistance (R_{DS}(ON)) and high maximum current ratings. For example, the SPP08P06P has an on-resistance of 0.3W at 8.8A of continuous drain current for a cost of less than a dollar. Lower resistances and higher current ratings are available, as well.

For more information about using PMOS and NMOS devices for polarity protection, see Maxim Electronics Application Note 636, *Reverse Current Circuitry Protection.*[3]

Relay Protection

Figure 3 shows a pair of circuits that can be used for polarity protection. Relay-based circuits have the advantage of little to no voltage drop — even at high currents — as long as the contact ratings are sufficient. The circuits are more



Diodes are 1N4001
Capacitors are 100 pF - .01 µF

FIGURE 3. Relays can be wired to open (A) or not close (B) when reverse polarity voltage is applied. The diode across the relay coil clamps the kickback voltage, and the capacitor prevents RF rectification and EMI generation.

complex than diode-based circuits, but can generally handle more current and are not damaged by reversed polarity. The circuits reset themselves automatically.

The circuit in **Figure 3A** uses the normally closed (NC) contacts of a single pole, double throw (SPDT) relay. There is no current drain through the relay coil until reverse polarity voltage is applied. There will be a few milliseconds, however, during which reverse polarity voltage gets through to your circuit if no power switch is used or if the power switch is closed.

This is generally enough time for damage to occur, so this circuit is only recommended if a power switch is used to turn the equipment on and off.

Normally open (NO) contacts are used in **Figure 3B**. The relay contacts close and supply power to the equipment only when applied voltage has the proper polarity. The relay coil draws current continuously during normal operation.

Small relays draw anywhere from 10 to 50 mA, and this level of current drain may be unacceptable for low power and battery powered equipment.

Dealing with Transients

A transient, by definition, "... refers to momentary over-voltages or voltage reductions in an electric power system ..." (CRC Electrical Engineering Dictionary). Just because a transient is momentary, doesn't mean you can ignore them. Transients can cause a lot of damage!

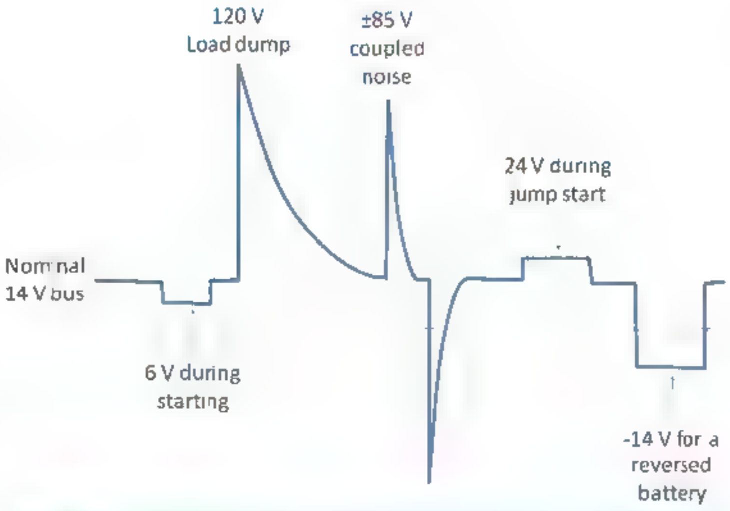


FIGURE 4. Typical vehicle voltage levels showing the different conditions present in an automotive power system (*based on Littelfuse reference 4*).

Automotive Transients

You might think that being supplied by a battery, a vehicle's DC power system is reasonably well-behaved. Unfortunately — from an electronics view — that's not true (as you can see in **Figure 4**). The nominal 14V DC source varies between 10.5V (a discharged battery) to more than 15V during heavy charging, and even 24V during a jump-start. The typical input voltage specification of 13.8V ±15% for equipment designed to be used in a "12 volt" vehicle corresponds to a fresh battery during normal charging.

In addition to DC voltage variations, there are numerous transients superimposed on the system. For example, rapid current switching in the vehicle's alternator creates sharp transients on the power bus (the heavy wiring that distributes power within the vehicle).

If adequate filtering is not performed by radio or audio equipment, the result is an audible, high-pitched alternator whine that follows engine speed.

SAE standard J1113 — *Immunity to Conducted Transients on Power Leads* — (SAE is the Society of Automotive Engineers) describes these common transients encountered on a vehicle's power bus:

- Load Dump — Occurs when a loose battery connection opens up during charging and the alternator's energy is "dumped" on the power bus with no battery to hold down the voltage.
- Alternator Field Decay — Occurs every time the vehicle is turned off and the alternator's stored energy has to be dissipated via the power bus.
- Inductive Load Switching — The kick-back voltage from an inductive load (like an electric window motor) being turned off.
- Mutual Coupling — Transient energy that is coupled between conductors in a wiring harness.

Table 1 summarizes the electrical characteristics of these transients. There are quite a few more described in the standard, but these cover the range of

References

1. T. Fletcher WAØITP, "Reverse Voltage Protection Overview," *The QRP Quarterly*, Spring 2012, pp 12-13. Also online at **www.wa0itp.com/revpro.html**.

2. *The ARRL Handbook for Radio Communications*, 2016 Edition. Available from booksellers or from **www.arrl.org/shop**.

3. **www.maxim-ic.com/app-notes/index.mvp/id/636**

4. Littelfuse, *Suppression of Transients in an Automotive Environment*, AN9312.5, 1999, **www.littelfuse.com/data/en/Application_Notes/an9312.pdf**.

5. Littelfuse, *ESD Protection Design Guide*, **www.littelfuse.com/data/en/Applications/Littelfuse_ESD System_Level_Guide.pdf**.

6. Littelfuse, *Tips for Enhancing ESD Protection*, **www.littelfuse.com/data/en/Application_Notes/Littelfuse-Tips_for_Enhancing_ESD_Protection.pdf**.

7. R. Standler, *Protection of Electronic Circuits from Overvoltages*, Dover, 1989.

Table 1 — Typical Vehicle Transients.

Type	Voltage	Energy (Joules)	Duration	Occurs
Load Dump	<125V	>10 J	200-400 msec	Infrequently
Field Decay	-100V to +40V	<1 J	200 msec	At turn-off
Inductive	-300V to +80V	<1 J	<320 µsec	Often
Mutual Coupling	<200V	<1 J	1 msec	Often

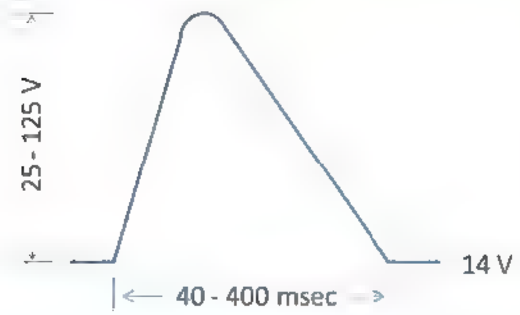


FIGURE 5. The load dump transient that occurs when the connection between a battery and the alternator opens during charging. Peak voltage can be more than 100V and the transient can last for nearly 1/2 second (*based on Littelfuse reference 4*).

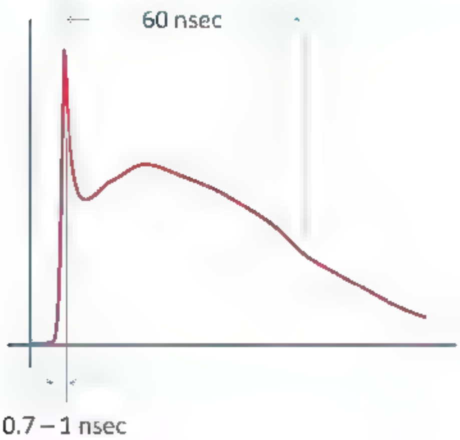


FIGURE 6. A typical ESD transient waveform specified by standard IEC 61000-4-2. Peak voltage from an ESD transient can reach 50 kV. Pulse voltages can be either positive or negative (*based on Littelfuse reference 6*).

voltages and energies vehicle electronics have to deal with.

Obviously, some of those transients are pretty severe — particularly the big load dump transient in **Figure 5**. It's a wonder anything survives being plugged into a vehicle! Your electronics are not going it alone, however, as the vehicle's manufacturer has built in some transient protection for you.

As explained in the Littelfuse application note, *Suppression of Transients in an Automotive Environment*, vehicle electronics are already protected by a central suppressor in the vehicle, usually located close to the master control computer module.[4]

There are usually suppressors in other modules around the vehicle, too. This helps limit the voltages and energies to which your gadgets and radios are exposed. Nevertheless, it's a good idea to provide some limited protection by using protective components on input and output connections.

Electrostatic Discharge and Lightning

Another source of transients — well known to inhabitants of areas with dry weather — is electrostatic discharge, or ESD. A sudden discharge of static electricity by a spark — such as from walking across a carpet and then touching a grounded surface — is a typical example of ESD.

In fact, the standard ESD test generator uses a finger-shaped probe. There can be enough energy in an ESD pulse to destroy semiconductors or scramble the operation of a circuit.

A typical ESD transient lasts for less than a microsecond (see **Figure 6**), but can generate voltages up to 15 kV! ESD transients can appear on power and signal wiring, connectors, controls and switches, displays — anything a finger can touch or approach ... even metal enclosures. Because the pulses are short, the most effective strategy is to use an RC circuit to slow down the voltage rise as in **Figure 7**.

Direct hits from lightning can be devastating, but even an indirect or nearby strike can generate pretty significant transients. These fast-rising transients have energies equal to or larger than the automotive inductive transient, and with voltages nearly as high as an ESD pulse.

Hams having erected towers and antennas have out of necessity developed a wide range of techniques dealing with lightning's effects — ranging from grounding and bonding techniques through voltage clamping circuits.

A number of tutorials and related articles and standards are available at **www.arrl.org/lightning-protection**.

Protective Components

Both input and output connections from your equipment need protection from lightning and ESD. Depending on your budget and the power levels involved, there are a number of different components you can use.

Low power protection can be as simple as an RC low pass filter to damp and absorb transients.

At higher power levels, you'll need to use more robust components and possibly apply several different techniques to provide overlapping layers of protection. It is not unusual to find clamping diodes, EMI and ESD filter capacitors, and a beefy MOV all connected to the same circuit!

The following components give you several ways to protect electronics against transients — block them, route them away from the circuitry, and dissipate their energy as heat. The goal is to limit the resulting voltage to levels the electronics can handle.

The cost for these components is quite modest and purchasing in quantity makes them easy to apply:

- MOV (Metal Oxide Varistor; **en.wikipedia.org/wiki/Varistor**) — Partially conductive powder pressed into a disc or cylinder so that it is non-conductive up to its clamping voltage. At voltages (of either polarity) higher than the clamping voltage, MOV resistance drops, limiting the voltage by dissipating energy as heat.

MOVs are generally connected between the circuit being protected and ground so that the lower resistance causes the MOV to absorb the transient's energy and keep voltage at a safe level.

After repeated transients, MOVs generally fail in a low resistance state. The V22ZA05P is a typical small MOV that clamps at 18.7V and can dissipate 0.2J.

- TVS Diode (Transient Voltage Suppressor; **en.wikipedia.org/wiki/Transient_voltage_suppression_diode**) — A special type of zener diode that offers more precise clamping action. Two TVS diodes back-to-back in a single package make a bipolar clamp similar to an MOV. Unless overloaded, TVS diodes can handle repeated transients without changing their characteristics.

The 1.5KE11A is a bipolar device that clamps voltage at ±15.6V and can dissipate 1,500W for 1 msec.

- Diode Clamps — Most effective for ESD and fast transients, clamping diodes route energy away from the protected circuit into the power supply where it is absorbed by the filter components.[5,6]

The clamping diode is connected so that current can flow from the circuit to be protected into the power supply circuit where the energy is absorbed by the filter capacitors. This limits the circuit voltage to one forward voltage drop beyond the power supply voltage.

Many ICs used for input/output functions have small clamping diodes built in at the I/O pins. For low power signal lines, the common 1N4148 will do a good job.

- RC Pulse Filtering — Useful for both RFI and ESD, a series resistor followed by a capacitor to ground acts as a low pass filter, smoothing out transient waveforms to limit voltages and currents. Clamping diodes are often used as well, as in **Figure 7**.

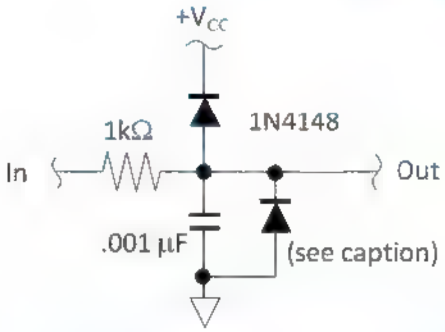


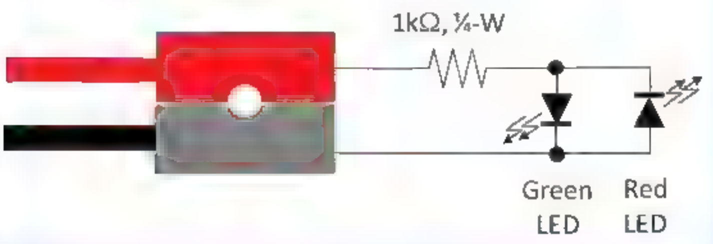
FIGURE 7. A simple low pass RC circuit helps protect inputs against ESD pulses. The negative voltage clamping diode's anode should be connected to the negative power supply, if available.

That's a Wrap

This short discussion just scratches the surface of transient protection. The referenced application notes can provide a lot more information — Littelfuse offers many application notes on its website, and references like Standler's book on transients is excellent and detailed.[7]

In the meantime, protect your circuits and equipment — especially if used in vehicles or away from the safety of the home lab! **NV**

FIGURE A.



A handy voltage polarity checker can be made inexpensively and added to your "go" kit. Although Anderson PowerPoles are shown here, the circuit will work with any connector style.

Verify that Voltage!

Assuming voltage polarity is okay if the power source has a red and black connection can be an expensive assumption. It's always a good idea when connecting your gadget (or radio) to an unknown power source to first check for proper voltage polarity. **Figure A** shows a simple circuit you can make and use with any connector style, although the current ham radio standard, Anderson PowerPoles is shown at **www.andersonpower.com/us/en/products/powerpole/index.aspx**. No enclosure is necessary and hot glue or epoxy can be used to secure the small components right on the connector.

READER - TO - READER TECHFORUM

All New TECHFORUM Online At
www.nutsvolts.com/tech-forum

>>> QUESTIONS

Super Capacitor Comparison

I need some help in understanding what's going on inside a super capacitor. I did some experiments using regular capacitors as a backup power supply to a real time clock chip. Mathematically, the amount of time an electrolytic capacitor (1500, 2200, and 4700 µF) would power the chip became predictable once I came up with a formula. However, when I connected a super capacitor, the math broke down.

The real time clock should have exhausted the stored power in the super capacitor after exactly three days. Instead, it is still maintaining the correct time after four months! Clearly, something is physically different about a super capacitor. It seems to be acting more like a battery than a capacitor.

Can someone explain to me how the chemistry of a super capacitor differs from electrolytic capacitors? Why does the amount of charge stored seem to far exceed the capacity indicated by its Farad value, under an extremely light load?

#11151 **Judy May W1ORO**
Union, KY

Controller Quandary

I'm stuck with the limitations of my controller which has eight analog inputs each which can sense 0.5V changes between 0-10V. I need to measure temperature between 50-160°F within one or two degrees.

My idea to get the accuracy needed is to divide the thermistor output across three inputs where input one would resolve the 100's, input two would resolve the 10's, and three would be the 1's. Example: temp 143 divided into three would produce a 1V signal on input one, a 4V signal on input two, and a 3V signal on input three. Then, in software in the controller, recombine the values back into a single temperature.

Does this concept seem doable and if yes, what would be the easiest way to create such a circuit? Thanks in advance for any assistance you can provide.

#11152 **Harvey Mushman**
Los Angeles, CA

>>> ANSWERS

### [#8151 - August 2015]
Headphone Hookup

My flat panel TV doesn't have a headphone jack. It has dual RCA jacks labeled "R/L line out" and a "TOSLINK" connector. What would be the simplest way to get my headphones hooked up to this TV?

I would use an external computer speaker set. It will provide a cheap small amplifier, volume control, and an input for your headphones to plug into. If it comes with an external power supply, you won't need to buy batteries. The computer speaker system will most likely come with a stereo mini 1/8" phono plug, so you will also need to get an RCA adapter cable. The computer speakers must have a place for you to plug in your headphones.

Attach the adapter to your TV and then plug the speakers into the adapter. Once the speakers are connected, you should be able to play audio out of them. Then, plug in your headphones and the speakers will shut off and all the sound will feed into your headphones. Adjust the volume on the speakers.

Done and your total cost should be less than $25 ($15-$20 speakers and $5 for the adapter).

Harvey Mushman
Los Angeles, CA

### [#9151 - September 2015]
Speaker Re-Coning

I have an older pair of Cerwin-Vega speakers where the foam edge has rotted and fallen apart on the woofers. I would love advice on the pros and cons of "re-coning" vs. buying new woofers. The model of the speaker is "R-24."

I was recently given an old pair of Cerwin-Vega speakers from a co-worker in much the same condition, it sounds, as yours are. The woofers in both cabinets were electrically good as was the physical condition of the voice coils and cones. Both woofers, though, exhibited the common problem of foam rot. Not only does the foam separate but seems to dissolve leaving the cone and voice coil to rattle around without direction.

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES WHATSOEVER** are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

I performed a web search on the subject and found a solution that seemed very reasonable. Simply Speakers (**simplyspeakers.com**) has re-foam kits for numerous speaker manufacturers including Cerwin-Vega. I'm not touting Simply Speakers as the only provider, but they were excellent to work with, sent me everything needed to do the job, and provided a YouTube video on how to do it. Search YouTube for "Simply Speakers" to see the video. I just finished mine last week with excellent results.

Mike Hudson
Boise, ID

[#10151 - October 2015]
Variable Speed Motor

I'm looking to purchase a belt grinder for my workshop to smooth the edge of circuit boards, aluminum cutouts, and the output of my 3D printer. My question relates to the variable voltage controllers and three-phase motors that are available as optional equipment. Given almost double the price of the grinder, can I simply use a surplus variac on a single-phase motor to get a variable speed grinder?

The speed of an induction motor is dependent primarily on the frequency of the alternating current driving it. Speed controls for three-phase induction motors are often called VFDs which stands for Variable Frequency Drive. They work by converting the incoming AC power to DC and then use a three-phase inverter to convert it back into AC with the ability to vary the frequency from as low as a few Hz to 120 Hz or more for motors that can tolerate the increased speed.

If you attempt to reduce the speed of an induction motor by controlling the voltage with a variac, you will likely overheat the motor without it slowing down much, because the current will increase as you reduce the voltage. Inexpensive variable speed power tools are powered by universal motors that have brushes and a commutator and these can be controlled with a Variac, though the common triac light dimmer type circuit is far more economical.

If you can find a grinder powered by one of these, then your idea will work, but most bench mounted tools use induction motors.

James Sweet
via email

[#10152 - October 2015]
ConFused

I seem to go through fuses quickly on my bench power supply. Would it be okay to try a higher than normal value fast-blow fuse or go with a slow-blow fuse of the original value?

It's not a good idea to over-fuse your power supply (PSU). The fuse was designed to protect the supply from damage (and fire) if it becomes defective internally or is operated beyond its design limits. You are probably exceeding its capabilities in some way, such as overcurrent due to a too heavy load. If you're not exceeding its specifications, there might be something wrong inside the PSU, such as bad filter capacitor(s) or a defective power transformer.

Have you checked the output under load with a scope? That will tell you if the PSU has high ripple under load — an indication of poor filtering inside the PSU. I suggest that you do a little investigation to determine whether the blown fuses are due to trying to operate it beyond its capabilities or bad component(s) in the PSU.

Your multimeter is a good tool to help do this. Watch the needle or display and see what the output voltage does right before the fuse blows. Analog meters are better in this situation. The first thing that comes to mind; does your load have a large capacitor that needs to be charged by the PSU? Large capacitors need high values of surge current from the supply until they acquire a full charge. If that's the case, you might lower the value of the capacitance at the load. Use your multimeter as an ammeter to watch the current to the load. Is it at or beyond the specified rating of the PSU? Again, an analog meter is best.

If you're operating the PSU at its limits, the internal circuitry could be overheating. Mount a fan or blower so that it directs air over the heat-producing components (heatsink, power transistors, power transformer). If you absolutely need to run the PSU at its limits, you might consider getting a more robust PSU. It will be more likely to survive.

Cheap PSUs are sometimes over-specced, meaning that they meet specs only under very controlled conditions. Also, is the PSU rated for full output continuously? It might be overheating if it's not rated for continuous operation.

The last thing I can suggest is to check your mains voltage to the PSU. Is it at or near the high limits of the PSU? If so, you might use a variac or bucking transformer to lower the mains voltage to the supply.

Hope that gives you some ideas that will help determine why the fuses are blowing so frequently.

David Mason
Hazel Green, AL

READER FEEDBACK

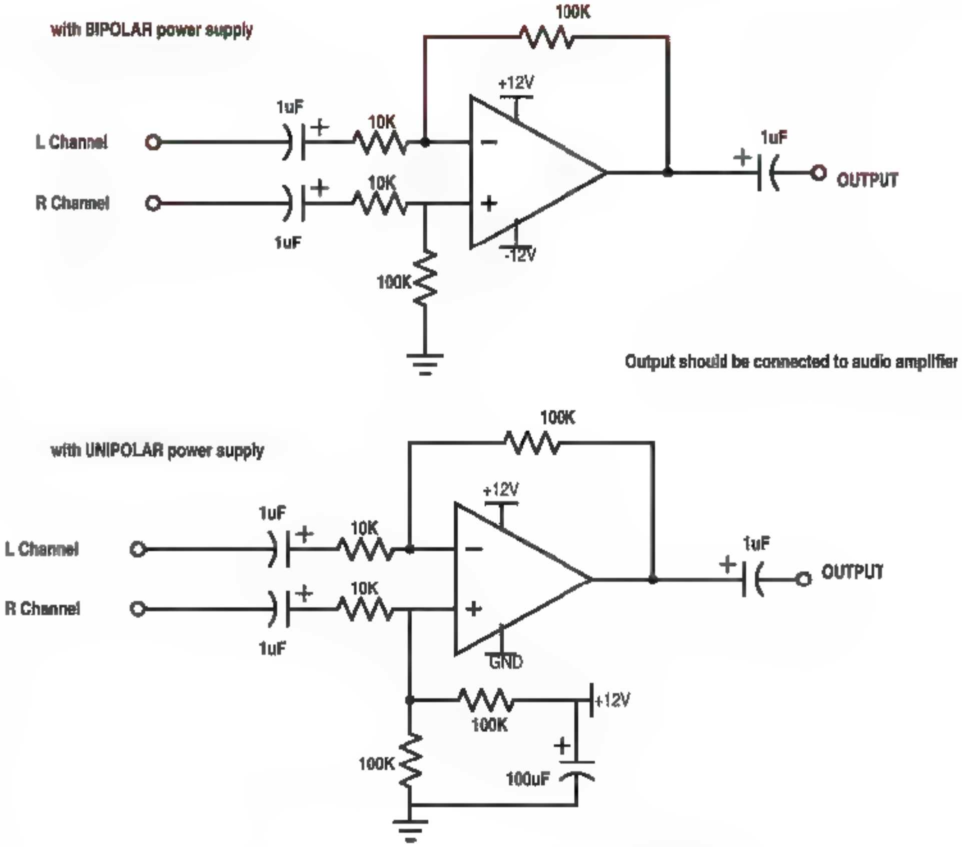
Continued from page 7

I wanted to point out a small error. In Figure 9 on page 19, there is a 100 µF capacitor incorrectly connected to the non-inverting (+) input of the op-amp to ground. This configuration is okay, but only if the inverting input is to be used. However, in the diagram, it shows both inverting (-) and non-inverting (+) inputs being used in this circuit. The 100 µF capac tor to ground is essentially shorting any audio to ground from the non-inverting (+) input.

Solution: Connecting the 100 µF capacitor on the +12V DC side would be the correct placement. Perhaps the author originally intended this.

I believe the schematic included here will correct the problem. EngineersGarage originally posted the schematic and I didn't take time to analyze it properly. At 3,000 Hz, the capacitive reactance is only around 0.5 ohms, so the right channel would have been short-circuited to ground. Thanks for catching my senior moment.

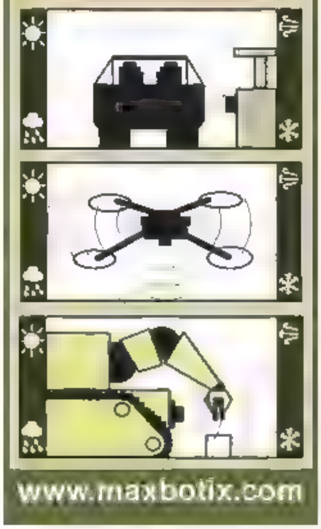
Tim Brown Q&A



VOICE EXTRACTOR

adapted from Engineers Garage (http://www.engineersgarage.com/contribution/voice-extractor-circuit#)

CLASSIFIEDS



Continued from page 25

- Category 1 receivers ensuring maximum compliance for safety applications.
- Four second local mode alarm output trigger (60 second full system).
- Battery powered — very low power consumption.
- Lost units can be detected and reported via serial port.
- Serial output on each board which could be connected to a PC or alarm panel for greater functionality (a conversion board to allow communications in a format for any alarm system is available).

The basic system has 40 nodes. However, this can be expanded to another 10 (can have more, but not recommended) satellite units attached to each node.

The satellite units are given the same number as the node, so they cannot be distinguished as individual units for the purpose of identification; they allow an 'area' to be detected.

A typical use of this would be in a fire alarm where you wanted smoke detectors around an area (500 m radius) around a node, which then communicated on to the rest of the network. Satellites are mainly used to 'report in' an alarm.

There is also an option for 'listen only' units which can be positioned anywhere in the system. If there is an alarm, they will sound if a sounder is connected to the output. These units are called drones.

A drone could be used as a display unit if an optional display board is attached.

A system consists of a main board A40 and a display board A40D The main board can be configured as:

Master — numbered 00 — one per system. This is the main unit giving out timing pulses, but is also a working unit.

Node — numbered 01 -39. These are the backbone of the system enabling meshing; drones and satellites do no mesh.

Drone — numbered 41. These are receive only and are used as sounders or perhaps with displays attached.

Satellite. These are attached to nodes to expand the system and are numbered the same as the node, with a special link across two pins to enable this mode. The display can be attached to any of the above to enable it to display status.

Contact Lemos for pricing.

For more information, contact:
Lemos International
www.lemosint.com

NEW LCC SOCKET DESIGN

Ironwood Electronics has recently introduced a new LCC socket design using high performance elastomer capable of 75 GHz, very low inductance, and wide temperature applications. The GT-QFN-3010 socket is designed for a 14.3 x 14.3 mm package size and operates at bandwidths up to 75 GHz with less than 1 dB of insertion loss. The socket is designed to dissipate a few watts using a compression screw, and can be customized up to 100 watts with a heatsink on top of the screw and by adding an axial flow fan. The contact resistance is typically 20 milliohms per pin. The socket is mounted on the target PCB with no soldering, and uses very small real estate allowing capacitors/resistors to be placed close by. The socket is constructed with a swivel lid which incorporates a quick insertion method so that ICs can be changed out quickly. The socket components are gold plated for RF shielding.

Pricing for the GT-QFN-3010 is $978 at qty 1; reduced pricing is available depending on quantity required.

For more information, contact:
Ironwood Electronics
www.ironwoodelectronics.com

NEW MILLING MACHINE

Tormach, Inc., has announced their newest entry in personal CNC machinery: the PCNC 440. This CNC mill takes the design philosophies of Tormach's other machinery and makes them fit in an even smaller package.

The PCNC 440 has a footprint of 40" x 32" x 42" (X x Y x Z) and weighs just 450 lbs, yet boasts a spindle speed of 300 to 10,000 RPM, a max feed rate of 135 IPM (X,Y), and the ability to cut everything from plastics to aluminum to harder metals like steel and titanium.

It is controlled by PathPilot — Tormach's control system that is used on every PCNC mill.

Tormach's new PCNC 440 starts at $4,950. PCNC 440s will start shipping in November 2015.

For more information, contact:
Tormach
www.tormach.com

■LOOK FOR ■SEARCH FOR ■FIND
Find your favorite advertisers here!

ADvertiser INDEX